

RICE UNIVERSITY

**A Numerical Study of an Adjoint Based Method  
for Reservoir Optimization**

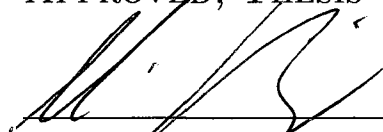
by

**Klaus D. Wiegand**

A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE

**Master of Arts**

APPROVED, THESIS COMMITTEE:



---

Dr. Matthias Heinkenschloss, Chairman  
Professor of Computational and Applied  
Mathematics



---

Dr. Mark Embree  
Professor of Computational and Applied  
Mathematics



---

Dr. Tim Warburton  
Associate Professor of Computational and  
Applied Mathematics

HOUSTON, TEXAS

APRIL, 2010

UMI Number: 1486045

*All rights reserved*

**INFORMATION TO ALL USERS**

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1486045

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

## **Abstract**

# **A Numerical Study of an Adjoint Based Method for Reservoir Optimization**

by

Klaus D. Wiegand

A numerical reservoir simulator that uses a finite volume spatial discretization and two time discretization schemes is developed and tested. First and second order derivatives for the numerical simulator are derived, using an adjoint based approach. The adjoint and derivatives are validated in the context of the time discretization schemes, using varying time step sizes, and compared for accuracy. Two optimization algorithms are developed and used in combination to solve a numerical reservoir optimization problem. Numerical results are presented and discussed.

## Acknowledgements

I thank my advisor Dr. M. Heinkenschloss for his support and guidance in this research. I also thank Dr. M. Embree and Dr. Warburton from the CAAM department for supporting my research and evaluating this thesis. I am indebted to my friend Dr. Amr El-Bakry for his long standing support and interest in my work. Finally, I thank my wife Linda for her patience while I tried to balance my work-life, research, and my obligations to my family.

My research was supported by ExxonMobil Upstream Research Company. I am deeply indebted and grateful for the financial and moral support that I received from my employer.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Organization of the Thesis . . . . .	3
<b>2 Literature Review and Problem Statement</b>	<b>5</b>
2.1 Numerical Reservoir Simulation . . . . .	5
2.2 Numerical Reservoir Optimization . . . . .	6
2.3 Research by Other Authors . . . . .	8
2.4 Contribution of my Thesis . . . . .	16

<b>3</b>	<b>Numerical Reservoir Model</b>	<b>18</b>
3.1	Rock Model . . . . .	18
3.2	Fluid Model . . . . .	22
3.3	Transport Model . . . . .	23
3.4	Equations for incompressible Flow . . . . .	23
3.5	Finite Volume Discretization . . . . .	26
3.6	Time Discretization . . . . .	33
3.6.1	IMPSAT . . . . .	34
3.6.2	SEQUENTIAL . . . . .	35
<b>4</b>	<b>Optimal Well Rate Allocation</b>	<b>38</b>
4.1	Optimal Control Problem . . . . .	39
4.2	Adjoint and Gradient Computations . . . . .	41
4.2.1	Gradient Computation for an Abstract Problem . . . . .	41
4.2.2	Gradient Computation for Semi-Discretized Case . . . . .	44
4.2.3	Gradient Computation IMPSAT . . . . .	47
4.2.4	Gradient Computation SEQUENTIAL . . . . .	50
4.3	Second Order Derivatives . . . . .	55
4.3.1	Second Order Derivatives for Abstract Problem . . . . .	55
4.3.2	Hessian times Vector Product for Reservoir Problem . . . . .	57
4.4	Numerical Experiments for Adjoints . . . . .	60
4.4.1	Reservoir Model . . . . .	60

4.4.2	Experimental Setup . . . . .	62
4.4.3	Discussion of Numerical Results . . . . .	67
4.4.4	Conclusions . . . . .	70
<b>5</b>	<b>Numerical Optimization</b>	<b>71</b>
5.1	Introduction . . . . .	72
5.2	Optimization Algorithm . . . . .	75
5.2.1	Algorithm Overview . . . . .	75
5.2.2	Computation of $s_k$ . . . . .	79
5.2.3	Computation of $q_k$ . . . . .	85
5.2.4	Direction of Negative Curvature $d_k$ . . . . .	89
5.2.5	Computation of $p_k$ and Line Search Strategy . . . . .	90
5.2.6	Computation and Update of the Working Set . . . . .	91
5.3	Strategy for Problems with Large Working Sets . . . . .	92
5.3.1	Gradient Projection Method . . . . .	92
5.3.2	Integrated Optimization Approach . . . . .	97
5.4	Numerical Optimization Results . . . . .	98
5.4.1	Experiment 1 . . . . .	98
5.4.2	Experiment 2 . . . . .	107
5.4.3	Experiment 3 . . . . .	115
5.5	Discussion of Optimization Results . . . . .	123

<b>6</b>	<b>Conclusions and Further Work</b>	<b>125</b>
	<b>Bibliography</b>	<b>127</b>



# List of Figures

3.1	Relative Phase Permeability . . . . .	21
4.1	SPE10 Permeability Volume . . . . .	61
4.2	SPE10 Top Layer Permeability Distribution - $\log \left( \frac{K_{xx}+K_{yy}}{2} \right)$ . . . . .	61
4.3	Adjoint Experimental Setup - Well Placement . . . . .	62
4.4	Adjoint Experimental Setup - Injection Profiles . . . . .	63
4.5	Comparison of Gradients (Injectors 1,2) IMPSAT & SEQUENTIAL .	64
4.6	Comparison of Gradients (Injectors 3,4) IMPSAT & SEQUENTIAL .	65
4.7	Comparison of Gradients (Producers 1,2) IMPSAT & SEQUENTIAL	66
4.8	Comparison of Gradients (Producers 3,4) IMPSAT & SEQUENTIAL	67
4.9	Lateral shift of Gradient for Injector 3 . . . . .	68
4.10	Relative Difference in Adjoint - IMPSAT and SEQUENTIAL . . . . .	70
5.1	Optimization Experimental Setup 1 . . . . .	99
5.2	KKT Norm/Iteration - Experiment 1 . . . . .	102
5.3	Step Norm/Iteration - Experiment 1 . . . . .	103

5.4	Objective Function Values - Experiment 1 . . . . .	104
5.5	Computed Optimal Well Rates - Experiment 1 . . . . .	105
5.6	Final Aqueous Saturation Profile - Experiment 1 . . . . .	106
5.7	Optimization Experimental Setup 2 . . . . .	107
5.8	KKT Norm/Iteration - Experiment 2 . . . . .	110
5.9	Step Norm/Iteration - Experiment 2 . . . . .	111
5.10	Objective Function Values - Experiment 2 . . . . .	112
5.11	Computed Optimal Well Rates - Experiment 2 . . . . .	113
5.12	Final Aqueous Saturation Profile - Experiment 2 . . . . .	114
5.13	Optimization Experiment 3: Setup & Permeability Distribution . . .	116
5.14	KKT Norm/Iteration - Experiment 3 . . . . .	119
5.15	Step Norm/Iteration - Experiment 3 . . . . .	120
5.16	Objective Function Values - Experiment 3 . . . . .	121
5.17	Computed Optimal Well Rates - Experiment 3 . . . . .	122
5.18	Final Aqueous Saturation Profile - Experiment 3 . . . . .	123

# List of Tables

5.1	Iteration Results for Gradient Projection Algorithm - Experiment 1 . .	100
5.2	Iteration Results for Active Set Newton Algorithm - Experiment 1 . .	101
5.3	Iteration Results for Gradient Projection Algorithm - Experiment 2 . .	108
5.4	Iteration Results for Active Set Newton Algorithm - Experiment 2 . .	109
5.5	Iteration Results for Gradient Projection Algorithm - Experiment 3 . .	117
5.6	Iteration Results for Active Set Newton Algorithm - Experiment 3 . .	118

# Chapter 1

## Introduction

This thesis presents the results of my research in developing an adjoint based optimization method for numerical reservoir optimization. My work consists of three parts: First, I develop a numerical reservoir simulator for two phase, incompressible flow. The numerical simulator is based on a finite volume spatial discretization, and uses two different time marching schemes. The two schemes differ in their level of implicitness, and are compared against each other.

Second, I derive an adjoint based formulation to compute first and second order derivatives for the numerical simulator. First order derivatives are derived for both time discretization methods and are compared for accuracy, using varying time step sizes. Second order derivatives are computed for the more implicit time discretization. Numerical validation methods are developed for the adjoints and derivatives.

Third, I develop two optimization algorithms to solve numerical reservoir optimiza-

tion problems, using the developed simulator and the adjoint based derivatives. The two algorithms are combined such that the first computes an initial guess for the second. The main contribution in the third part of my thesis is the development of a second order active set method of the Newton type. Based on the previous works of Forsgreen and Murray [6, 7], I develop a method for large scale optimization problems, where the direct computation of the Hessian matrix is infeasible, and where function and derivative computations are prohibitively expensive. The combined optimization algorithms are evaluated, using three case studies for numerical reservoir optimization.

## 1.1 Motivation

The motivation for my research is threefold: First, the rising costs for the exploration and production of hydrocarbon resources are driving the business need for numerical optimization in many areas of reservoir management. Second, the advances in numerical reservoir simulation, and the availability of cheap high performance digital computers makes it nowadays possible to apply numerical optimization to large field studies that were impossible to conduct just a decade ago. However, the main motivation for my studies is as follows: While numerical reservoir optimization is an active area of research in the petroleum industry and impressive results have been reported by some authors, many of the works I have studied in my literature review leave important questions unanswered, that I consider relevant. Such questions are of a basic

nature: What are the numerical properties of the optimization problem and its derivatives? What optimality conditions were used in a particular study, and were these conditions achieved? How were the computations (adjoints, derivatives, optimization results) verified? What conclusions can be drawn for the further development and refinement of optimization algorithms tailored towards reservoir optimization? Some of these questions are answered by certain authors, others are not. I therefore felt the need to conduct a basic study of numerical reservoir optimization that allows me to find more answers to these questions. I have done this in the context of a fairly simple numerical reservoir simulator that allows me to focus on these basic questions, and that also allows me to make certain necessary assumptions about the smoothness of the problem that I require for the mathematical derivation. My hope is that the presented work is flexible and extendable enough, as to apply it to more complex problems in my future research.

## 1.2 Organization of the Thesis

In the following chapter, I give an overview of where numerical reservoir simulation is used in the oil and gas industry, followed by a literature review of the current research in numerical reservoir optimization. I then define the optimal control problem I solve for my thesis work. Chapter 3 is devoted to the description of the discretized numerical reservoir model, followed by the derivation of the adjoint based derivatives in chapter 4. Numerical results depicting the impact of using either a fully implicit

or semi-implicit time stepping method are presented.

Chapter 5 develops the second order optimization algorithm that I use to solve the optimal control problem introduced in chapter 3. Numerical results for the optimization of three case studies are presented. The thesis closes with conclusions and suggestions for further research.

## **Chapter 2**

# **Literature Review and Problem Statement**

This chapter introduces numerical reservoir simulation and optimization, and presents the results of my literature review. I then describe the optimal control problem that I solve using the optimization algorithm developed in chapter 5.

### **2.1 Numerical Reservoir Simulation**

Numerical reservoir simulation is the simulation of fluid flow through hydrocarbon rich, subsurface reservoir rock, attached wells and surface facilities. The state of the reservoir (pressures, phase saturations, fluid composition, rock properties) is updated as the simulation advances in time, and fluid flow and pressure drops in attached wells and surface facilities are computed. Mathematically, the reservoir state has to



satisfy a set of coupled, nonlinear PDEs that are derived from physical conservation laws. Typically, a no-flow Neumann boundary condition is imposed at the perimeter of the reservoir, and time varying Neumann or Dirichlet conditions are imposed at locations where wells penetrate the reservoir rock. Mixed (Robin-type) boundary conditions can be found in simulators with thermal simulation capabilities, e.g., one can specify a bottom hole well pressure along with an assumed heat-loss in the well bore. Full physics reservoir simulators are used in the industry to realistically model both microscopic and macroscopic interactions of the fluid with the rock, as well as the behavior near wells. For the numerical study of discretization approaches, such as mesh generation and quality, time discretization schemes, or the study of optimization problems, simpler reservoir models are used that allow one to focus on the important aspects of the problem at hand.

## 2.2 Numerical Reservoir Optimization

The exploration, development and production of oil and gas fields is very capital intensive, and hence, process optimization can result in significant savings. There are three major stages in the project-lifecycle for a hydrocarbon resource.

### **Exploration Stage**

During this stage, assessments are made about the location, size, quality, and ultimate recovery of a reservoir. Geologic and seismic studies are conducted, drilling

cores and fluid samples are analyzed, and comparisons to fields similar in structure and size are drawn. The sparsity and uncertainty of the available data, coupled with economical and political uncertainties, favor optimization processes that evaluate return on capital under risk. Numerical reservoir simulation is commonly not applied during this stage. The work of Saito, de Castro, et al. [19] provide examples on how value assessments can be performed, however, their work and optimization during the exploration stage in general is not directly relevant to my thesis and is mentioned here for completeness.

### **Development Stage**

The development stage plans all major investments needed for the production of the field. This includes the number, size, and location of platforms; the number, type, and position of injection and production wells; the drilling schedule; infill positions for additional wells; and the construction of facilities such as hydraulic flow lines, pumps, compressors, separators, tanks, etc. The use of numerical reservoir simulators is common during this phase, and more recently, surrogate models have been used to evaluate possible development scenarios. The use of surrogates or reduced order models is attractive at this stage, since engineers try to establish a “big picture” view of possible development scenarios and don’t require all the details of a full physics simulation. Although my work is applicable in this area, the high level of uncertainty in the available data at this stage makes the use of simulation based optimization less attractive.

## Production Stage

During the production stage, hydrocarbons are extracted from the reservoir based on the planned depletion strategy. The day to day operations focus on controlling well rates, wellhead and flow line pressures, maintaining reservoir pressure, and managing operating constraints such as gas and water production, or injection constraints. Numerical reservoir simulation is used extensively during this phase. The research of production optimization is of great relevance to my thesis work, since it is there where nonlinear, PDE constrained optimal control problems are studied.

The three stages described above offer an abundance of optimization opportunities. In the following, I will describe the most common optimization problems that have been researched and published by other authors.

## 2.3 Research by Other Authors

In the following, I give an overview of the research conducted by other authors. Not all of these works are directly relevant to my thesis, but are listed to provide some closure.

### Optimal Well Placement

The optimal placement of wells has been one of the earliest attempts at reservoir optimization. Rosenwald and Green [18], as early as 1972, described the use of *Mixed Integer Programming* (MIP) to choose among  $N$  predefined potential well locations for

$M < N$  wells. Seifert et al.[21] describe a stochastic model to place wells based on an exhaustive analysis of potential well locations and well trajectories. The research of Ierapetritou, Floudas, et al.[9] builds on Seifert's earlier work, and combines a decomposition approach with MIP in individual regions.

Bangerth, Klie, Wheeler et al.[4] formulate the well placement problem as an unconstrained integer optimization problem, where well locations are selected from a set of candidate locations, and the objective is a discounted cost function that takes into account the revenue of produced oil and the cost for injecting and disposing water. In their formulation of the dynamic state equations for the reservoir model, well rates are determined by location and a fixed bottom hole pressure, defined for both injectors and producers. The authors evaluate and compare five optimization approaches with focus on robustness and efficiency. For problems involving multiple (7) wells, the authors obtain the best results in using a fast simulated annealing algorithm. I consider this approach very interesting, however, I have concerns that the method will become inefficient when trying to solve the problem for a large number of wells with many candidate locations.

Industrial numerical reservoir simulators use spatial discretizations, where wells are connected to nodes of finite difference grids, or to cells in finite volume grids. The actual physical location of the well perforation within a reservoir cell is used to compute the reservoir-well connectivity (r-value). The commonly used formula to compute r-values is differentiable, and was developed by Peaceman [15, 16]. Hence, the optimal

placement of wells can be treated as a continuous optimization problem, where the computed locations are translated into a node/cell index and the corresponding  $r$ -value. In my thesis work, the main effort has gone into the adjoint and derivative computation, as well as in the development of the optimization algorithm. Most of this work can be re-used when considering a different set of control variables, e.g., well positions.

### **History Matching**

History matching is a process where the reservoir model properties are updated in order to match simulation results with observed reservoir behavior. The problem is difficult for several reasons. First, it involves multiple scales, since large features, such as the location of faults, channels, or the depth and tilt of the oil-water contact have to be considered together with properties that vary on a much smaller scale, such as permeability and porosity. Second, it is an inverse problem with many local minima and sparse input. Third, the parameter space is highly dimensional.

History matching is still a largely manual process in most energy companies, but in recent years, attempts have been made to assist or automate the process. Of special interest for my thesis are the works of Aitokhuehi and Durlowski [2], and P. Sarma et al. [20]. Their research is targeted at finding solutions for the combined problem of production optimization and history matching in a feedback loop. I will discuss their research below.

## Production Optimization

Production optimization, especially in the form of optimal well control, is the area most relevant to my thesis work. During my literature review, I have selected seven research contributions that I like to discuss in more detail. Before I start the discussion, I want to point out the early work of Ramirez [17], who in 1987 published his pioneering book about production optimization. To my knowledge, Ramirez is among the very first who formulated the problem of production optimization in the context of an optimal control problem, using the adjoint approach to compute derivatives of the objective function. His work, in many ways, inspired me to seek a mathematical rigorous approach to solve the reservoir optimization problem posed in my thesis.

I start my evaluation with the work of Zandvliet et al. [25], who describe the application of Bang-Bang (on/off) well control to solve an optimal reservoir flooding problem. Their main result states that many reservoir flooding problems can be formulated as optimal control problems that are linear in the well controls, and that in the presence of simple bound constraints, these problems often have bang-bang optimal solutions.

In their research, Zandvliet et al. define their objective function to be:

$$J(u) \stackrel{\text{def}}{=} \int_0^T l_1(x(t), t) + l_2^T(x(t), t)u(t) dt, \quad (2.1)$$

where  $l_1, l_2$  are terms related to production and injection, and where  $x(t)$  represents the state vector, and  $u(t) \in \mathbb{R}^m$  is the vector of controls which is bounded from below and above. The objective function is similar in structure to the one I use in my work, with two differences: First, the controls enter the integrand in (2.1) in a

strictly linear fashion, and second, Zandvliet et al. restrict the control to injection, whereas I control both injection and production. The authors then define the optimal control problem to maximize (2.1), subject to state constraints, initial conditions, and the bound constraints on the controls. They show that the form of (2.1) imposes a particular structure on the first order necessary optimality conditions, that allows the optimal solution  $u^*$  to be defined in bang-bang form, that is:

$$u_i(t) = u_{min} \text{ or } u_i(t) = u_{max}, \quad i = 1, \dots, m, \quad t \in [0, T].$$

Zandvliet et al. don't provide details about the solution process, but state that they used a gradient based approach to solve their optimization problem.

I consider the approach interesting from a mathematical point of view, but of lesser practical applicability; the main reason being that shutting-in and re-opening wells is problematic in practice, especially when it is to be done frequently. In addition, the method is not extensible to more general problems because of its dependency on the special structure of the objective (2.1).

Wang et al.[23] in their work optimize the allocation of production well rates and gas lift gas rates (gas lift is used to enable/enhance the flow in production wells), using a sequential quadratic programming approach. However, their optimization is restricted to the gathering network of facilities, and does not treat the PDE based dynamic state equations of the numerical reservoir model as implicit constraints. Furthermore, the allocation optimization is not time dependent, but performed at each time step of the simulation process. Hence, their approach will be suboptimal

with respect to finding an optimal allocation pattern over the lifetime of the field.

For that reason, I decided to not adopt their approach.

Lorentzen et al.[10] propose the use of an Ensemble Kalman Filter as an optimization tool. Their main motivation for the approach is the complexity and computational cost associated with using a derivative based optimization method that considers the reservoir state equations. Instead, they assume an a priori knowledge of the net present value (NVP) or cumulative oil production over time, and use the Kalman filter to adjust control variables (chokes to regulate the flow between the well and the reservoir) that best match the predefined objective. The reservoir is treated as a black box

While their approach is an interesting and unusual application of the Kalman filter, it is unclear to me how optimality can be achieved or tested by their method (except for using a probably exhaustive search), and how constraints are handled.

The research of van Essen, Zandvliet et al.[22] is closely related to my thesis work.

The authors define the discretized optimization problem as:

$$\max_q J(q) \stackrel{\text{def}}{=} \max_q \sum_{k=1}^N L(x_k, q_k), \quad (2.2a)$$

$$s.t. \quad x_{k+1} = F(x_k, q_k), \quad x(0) = x_0, \quad (2.2b)$$

$$q_{min} \leq q_k \leq q_{max}, \quad (2.2c)$$

$$q_{o,k} + q_{w,k} = q_k, \quad (2.2d)$$



where  $F(x_k, q_k)$  denotes the PDE-based dynamic state equations of the numerical reservoir simulator,  $L$  is a measure of the reservoir performance, and  $q_{o,k}$ ,  $q_{w,k}$  denote the fractional flow of oil and water. The variable  $k$  denotes the time step number.

Van Essen, Zandvliet et al. describe the derivation of the gradient using an adjoint formulation, in a more general, abstract setting, and have used a gradient projection approach to ensure feasibility of the iterates. No details are provided about the performance of the optimization algorithm, what optimality criteria is being used, and the ability of their algorithm to identify the active set of inequality constraints at the solution. However, the main focus in their paper is the estimation of reservoir performance under uncertainty, which the authors try to address by applying the optimization to several different realizations of their numerical reservoir model.

The work of A. H. Althuthali, D. Oyerinde and A. Datta-Gupta [3] is also closely related to my thesis problem. The authors address the problem of using well rate controls to optimize the water flooding of a reservoir. However, the authors use a different approach, which I consider very interesting. The formulation of the objective function and its derivatives are based on time of flight and arrival computations for the water front that extends from the injectors to the producers. A fast streamline simulation approach or finite differencing is used to produce the necessary input data for the computation of derivatives. Wells are assembled in groups of expected equal arrival times to reduce the dimensionality of the problem and the number of simulation runs to compute the derivatives.

The approach seems very appealing to me, when considering sweep efficiency as the main problem to solve. However, my goal was to provide a more general framework that allows me to re-use many of the developed algorithms in the context of other reservoir optimization problems. Hence, I did not consider this approach for my thesis work.

The earlier work of Inegbenose Aitokhuehi and Louis J. Durlofsky [2] is a predecessor to the research conducted by Sarma, Durlofsky and Aziz [20], which is discussed below. The goal of the authors is twofold: Optimal well rate control for production optimization in combination with automated history matching. The optimization approach for the well rate control is very simple: A finite difference scheme is implemented to approximate the gradient of the objective function, with each gradient approximation requiring a forward simulation run. The computed gradients are used in a conjugated gradient optimization algorithm to determine optimal rates to maximize recovery.

The research of Sarma et al.[20] addresses the combined problem of optimal allocation control for water flooding and history matching. It is the first part of their work that is closely related to my thesis, and which is comparable to the works of van Essen, Zandvliet et al. The authors formulate the optimization problem as to maximize NPV over time and use an adjoint based approach to compute derivatives for the implicitly constrained optimal control problem. The adjoint approach is described in general terms; however, from attending a presentation at Stanford, I learned that

the approach was implemented in the context of a fully implicit time discretization scheme. No details are provided if and how the adjoint computations were verified. Sarma et al. claim that the problem can easily be solved using a steepest decent method, but do not provide details how feasibility is handled in their approach. It is open if an active set method or gradient projection is used and the optimality conditions are not clearly stated.

I consider the work of Sarma et al. very interesting, and in scope certainly much broader than the problem that I address in my thesis. However, many details with respect to the adjoint computation and optimization are unclear, and hence, it is difficult for me to fully evaluate their approach.

## 2.4 Contribution of my Thesis

I develop a rigorous mathematical approach to solve an optimal rate allocation problem, and evaluate the efficiency and accuracy of the method for fully implicit and semi-implicit time stepping schemes. Specifically,

- I develop a discretized numerical reservoir model for a shallow, incompressible, oil-water reservoir, using both fully implicit and semi-implicit time marching schemes.
- I then define the optimal control problem as a PDE constrained NLP.
- I develop an adjoint based method to compute first and second order derivatives.

- I develop a second order active set method to solve the optimal control problem.
- I report and evaluate numerical results for the adjoint based derivative computation and the optimization algorithm.

The following chapter describes the numerical reservoir model.

## Chapter 3

# Numerical Reservoir Model

In this chapter I develop the numerical reservoir model used in my research. The development follows Peaceman [14], as well as Aarnes, Gimse, and Lie [1], using a simplified 2-phase incompressible oil-water system with incompressible rock. The use of a simplified reservoir model is justified by my focus on the optimization problem, and the fact that full physics reservoir models exhibit strong discontinuities that make a rigorous mathematical approach almost impossible. Before describing the fundamental equations for fluid flow through porous media, I describe the components and assumptions that go into the model.

### 3.1 Rock Model

I assume incompressible rock with spatially varying porosity and permeability. Relative phase permeability is a nonlinear function of water saturation, that behaves

uniform throughout the reservoir. The following properties are loaded into the model.

### **Porosity**

Porosity describes the ratio of the pore space vs. the (bulk) volume of the rock. Physically, porosity is a scalar valued function of location, pressure, temperature, and time (due to hysteresis effects). In my model, porosity is a scalar valued function of space:

$$\phi(x) \in (0, 1), \quad x \in \mathbb{R}^2.$$

This rock model is called *incompressible*. Since porosity defines a ratio of volumes, it has no unit.

### **Absolute Permeability**

Absolute permeability is a tensor that defines the preferred flow direction of fluids inside the reservoir rock under influence of a pressure gradient. Most industrial reservoir simulators use a positive definite, diagonal permeability tensor in combination with a spatial discretization that aligns the outward normal of the mesh cell's interfaces with the flow direction. Such meshes are called k-orthogonal. I adopt this convention in my numerical reservoir model, using a rectangular grid. Specifically, the permeability tensor  $K$  is given by:

$$K(x) = \begin{pmatrix} K_1(x) & 0 \\ 0 & K_2(x) \end{pmatrix},$$

and where  $K_1(x), K_2(x) > 0, \quad x \in \mathbb{R}^2$ . The unit for permeability is the *Darcy*, which is equivalent to  $0.987 \cdot 10^{-12} m^2$ . The permeability of reservoir rock is usually small and

measured in milli-Darcy. However, in some natural reservoirs, absolute permeability can vary dramatically (5 to 7 orders of magnitude) on a small scale. This heterogeneity introduces strong jumps in the coefficients of the discretized reservoir model, creating difficulties for the numeric solution.

### Relative Phase Permeability

Relative permeability is a phase dependent fraction for absolute rock permeability. It is experimentally derived from laboratory data, and describes how one phase moves through the rock in the presence of another. Many rocks are *water wet*, meaning that water clings stronger to the rock surface than other fluids, allowing it to displace them. In oil-water systems, relative phase permeability is usually defined as a nonlinear function of aqueous saturation. The properties of natural reservoir rock often impose limits on phase saturation values due to microscopic effects. The *irreducible* oil and water saturations  $s_{or}$  and  $s_{wc}$  define lower bounds for the saturation values, and are set to  $s_{or} = 0.2, s_{wc} = 0.2$  in my model.

Let ‘ $l$ ’ denote the liquid hydrocarbon phase, and let ‘ $a$ ’ denote the aqueous phase.

Relative permeability is computed in my model as follows:

$$kr_l(s_a) = (1 - s^*)^2, \quad (3.1a)$$

$$kr_a(s_a) = s^{*2}, \quad (3.1b)$$

where

$$s^* = \frac{s_a - s_{wc}}{1 - s_{or} - s_{wc}}. \quad (3.1c)$$

The quadratic approximation is a reasonable model for relative permeability where hysteresis effects are ignored - see Peaceman [14, pp.14,15] for a reference. Since saturations add up to unity, the maximum water saturation is given by  $1 - s_{or}$ . The following figure depicts the functional dependence:

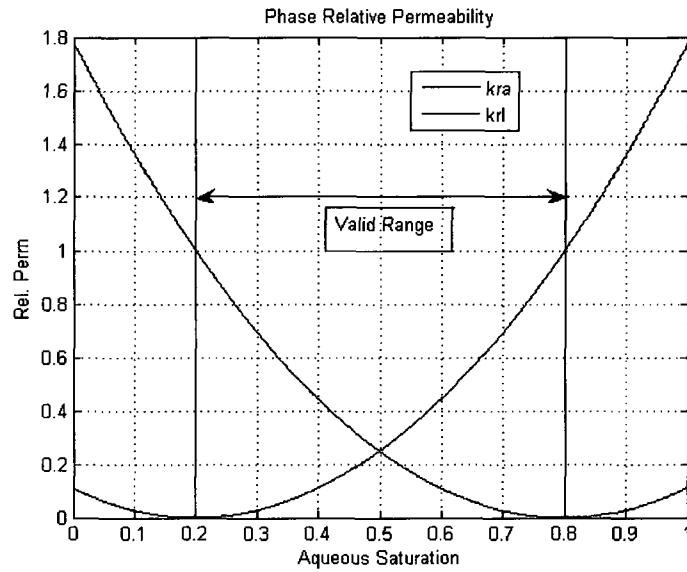


Figure 3.1: Relative Phase Permeability



## 3.2 Fluid Model

I consider an undersaturated *black oil* fluid model consisting of a liquid phase ( $l$ ) and an aqueous phase ( $a$ ) in water-wet rock. The fluid model has the following properties:

### Phase Densities

Phase density defines how much mass is associated with a given volume. The SI unit is  $kg/m^3$ . In my model, phase densities are normalized to unity,  $\rho_l = 1, \rho_a = 1 [kg/m^3]$ , since no additional insight for the optimization problem is gained by making them variable.

### Phase Compressibility

I assume a small *gas-oil* ratio for my undersaturated liquid hydrocarbon phase, and hence, I neglect phase compressibility. The aqueous phase contains only water and is assumed to be incompressible. Including or excluding liquid phase compressibility neither adds nor removes much complexity to the numerical model, however, incompressible flow models are commonly used in academic research (for example, see J. E. Aarnes and T. Gimse and K.-A. Lie [1]), and choosing this variant will allow a larger audience to compare their results with mine.

### Phase Viscosity

Physically, phase viscosity is a function of phase composition, pressure and temperature, however, commercial simulators usually hold viscosity constant over wide regions

of the reservoir model without losing accuracy. In my model, phase viscosity for the liquid and aqueous phase are held constant throughout the reservoir. The SI unit of viscosity is  $\text{Pa} \cdot \text{s}$ , however, the petroleum literature commonly uses the *centipoise*, where  $1\text{cp} = 10^{-3}\text{Pa} \cdot \text{s}$ . The values used in my model are:

$$\mu_a = 3 \cdot 10^{-4}, \mu_l = 3 \cdot 10^{-3} [\text{cp}].$$

### 3.3 Transport Model

Fluid flow through porous media at low to moderate velocities is governed by Darcy's Law, an empirical law discovered in the last century by Henry Darcy, a French engineer. For horizontal flow (thus neglecting gravitational effects), Darcy's law defines the following relationship between the phase velocity  $\vec{v}_\vartheta$ , rock and fluid properties, and an applied pressure gradient:

$$\vec{v}_\vartheta = -K \frac{kr_\vartheta}{\mu_\vartheta} \nabla p_\vartheta = -K \lambda_\vartheta \nabla p_\vartheta, \quad \vartheta = a, l \quad (3.2)$$

where  $\lambda_\vartheta \stackrel{\text{def}}{=} kr_\vartheta / \mu_\vartheta$  is the *phase mobility*.

### 3.4 Equations for incompressible Flow

The derivation of the differential equations describing incompressible flow is straightforward and well described in the literature. For a reference, again see Peaceman [14, p. 83]. The two fundamental equations are the pressure equation and the saturation

equation, which are both derived from the phase continuity equation and Darcy's law.

### Pressure Differential Equation

Let  $\Omega \subset \mathbb{R}^2$ ,  $\Omega$  open and bounded, be the region occupied by the reservoir. Under the assumptions stated in the rock and fluid model, the continuity equation for single phase flow is given by:

$$\phi(x) \frac{ds_{\vartheta}(x, t)}{dt} + \nabla \cdot \vec{v}_{\vartheta}(x, t) - q_{\vartheta}(x, t) = 0, \quad \vartheta = a, l, \quad x \in \Omega, \quad (3.3)$$

where  $q_{\vartheta}(x, t)$  is an arbitrary source/sink term. Summing over two phases yields:

$$\phi(x) \left( \frac{d}{dt} s_l + \frac{d}{dt} s_a \right) (x, t) + \nabla \cdot (\vec{v}_l + \vec{v}_a) (x, t) - (q_l + q_a) (x, t) = 0. \quad (3.4)$$

Saturations add up to unity, hence

$$s_l(x, t) + s_a(x, t) = 1 \text{ and } \frac{d}{dt} s_l(x, t) + \frac{d}{dt} s_a(x, t) = 0 \text{ for all } x \text{ and all } t.$$

This leaves us with:

$$\nabla \cdot (\vec{v}_l + \vec{v}_a) (x, t) - (q_l + q_a) (x, t) = 0. \quad (3.5)$$

Using Darcy's law (3.2) to substitute for the phase velocities yields:

$$\begin{aligned} & -\nabla \cdot K(x) (\lambda_l(s_a(x, t)) \nabla p_l(x, t) + \lambda_a(s_a(x, t)) \nabla p_a(x, t)) \\ & = (q_l + q_a) (x, t). \end{aligned} \quad (3.6)$$

For my incompressible fluid model, I can set  $p_a = p_l = p$ , Peaceman [14, p. 83]. Next,

I define the total mobility  $\lambda$  and combine the two source terms:

$$\lambda = \lambda_l + \lambda_a,$$

$$q = q_a + q_l.$$

Assuming that the reservoir is enclosed in an impermeable rock formation that inhibits communication of fluids with outside regions, I add a no-flow Neumann boundary condition, and arrive at the pressure differential equation:

$$-\nabla \cdot \lambda(s_a(x, t)) K(x) \nabla p(x, t) = q(x, t), \quad x \in \Omega \quad (3.7a)$$

$$\nabla p(x, t) \cdot \vec{n} = 0, \quad x \in \partial\Omega. \quad (3.7b)$$

### Saturation Differential Equation

Since saturations are required to add up to unity, it is sufficient to solve for the saturation of the aqueous phase. The saturation equation can be directly derived from the phase continuity equation (3.3) by substituting Darcy's law for the phase velocity, and adding boundary and initial conditions:

$$\frac{d}{dt} s_a(x, t) = \frac{1}{\phi(x)} (q_a(t, x) - \nabla \cdot \lambda_a(s_a(x, t)) K(x) \nabla p(x, t)), \quad t > 0 \quad (3.8a)$$

$$0 = \nabla p(x, t) \cdot \vec{n}, \quad (3.8b)$$

$$s_a(x, 0) = s_{a_{init}}(x). \quad (3.8c)$$

To summarize the above, the partial differential equations that govern the two-phase incompressible flow in my numerical reservoir model, together with their initial and boundary conditions, are given as:

$$-\nabla \cdot \lambda(s_a(x, t)) K(x) \nabla p(x, t) = q(x, t), \quad x \in \Omega \quad (3.9a)$$

$$\frac{d}{dt} s_a(x, t) = \frac{1}{\phi(x)} (q_a(t, x) - \nabla \cdot \lambda_a(s_a(x, t)) K(x) \nabla p(x, t)), \quad t > 0 \quad (3.9b)$$

$$\nabla p(x, t) \cdot \vec{n} = 0, \quad x \in \partial\Omega \quad (3.9c)$$

$$s_a(x, 0) = s_{a_{init}}(x). \quad (3.9d)$$

### 3.5 Finite Volume Discretization

For the finite volume formulation, we subdivide  $\Omega$  into  $N$  open subregions, such that:

$$\overline{\Omega} = \bigcup_{i=1}^N \overline{\Omega}_i, \quad (3.10a)$$

$$\emptyset = \Omega_i \cap \Omega_j, \quad i, j = 1, \dots, N, \quad i \neq j. \quad (3.10b)$$

Define  $\gamma_{ij} = \overline{\Omega}_i \cap \overline{\Omega}_j$ ,  $i, j = 1, \dots, N$ ,  $i \neq j$ , and for the  $i^{th}$  finite volume  $\Omega_i$ , let  $\mathcal{N}(i)$

denote the set of neighboring regions. In particular,

$$\gamma_{ij} = \emptyset, \quad j \notin \mathcal{N}(i), \quad (3.10c)$$

$$\gamma_{ij} \neq \emptyset, \quad j \in \mathcal{N}(i). \quad (3.10d)$$

### Spatial Discretization of the Pressure Equation

I now integrate (3.7a) over the finite volume  $\Omega_i$ :

$$\int_{\Omega_i} -\nabla \cdot \lambda(s_a(x, t)) K(x) \nabla p(x, t) dx = \int_{\Omega_i} q(x, t) dx. \quad (3.11a)$$

Under the assumption that  $\nabla p$  is continuously differentiable and that  $\partial\Omega_i$  is piecewise smooth, I apply the divergence theorem:

$$\int_{\partial\Omega_i} -\lambda(s_a(x, t)) K(x) \nabla p(x, t) \cdot \vec{n} dS = \int_{\Omega_i} q(x, t) dx, \quad (3.11b)$$

and using (3.10),

$$\sum_{j \in \mathcal{N}(i)} \int_{\gamma_{ij}} -\lambda(s_a(x, t)) K(x) \nabla p(x, t) \cdot \vec{n} dS = \int_{\Omega_i} q(x, t) dx. \quad (3.11c)$$

I now compute the flux across the interfaces  $\gamma_{ij}$  using a two-point flux approximation. For this, I approximate the pressure field by piecewise linear functions, with average cell pressures  $p_i$  defined at the center of the finite volumes  $\Omega_i$ ,  $i = 1, \dots, N$ , which are represented by the cells of a uniform rectangular mesh. I assume the cell interfaces  $\gamma_{ij}$  to be perpendicular to the principal flow direction imposed by the action of the diagonal permeability tensor  $K(x)$  on the pressure gradient. If the uniform distance between the cell centers of neighboring finite volumes is given by  $h$ , I can approximate the surface integral of the dot product for pressure gradient with the outward normal on the interface  $\gamma_{ij}$  by:

$$\int_{\gamma_{ij}} \nabla p(x, t) \cdot \vec{n} dS \approx \int_{\gamma_{ij}} \frac{(p_i - p_j)(t)}{h} dS, \quad (3.12a)$$

where the direction  $(p_i - p_j)$  is by choice. Using this approximation, (3.11c) becomes:

$$\sum_{j \in \mathcal{N}(i)} \int_{\gamma_{ij}} \lambda(s_a(x, t)) K_{ij} \frac{(p_i - p_j)(t)}{h} dS = \int_{\Omega_i} q(x, t) dx, \quad (3.12b)$$

where the sign change is introduced by my choice of direction  $\Delta p_{ij} \stackrel{\text{def}}{=} p_i - p_j$ , and where  $K_{ij}$  is the scalar interface permeability. I will describe shortly how  $K_{ij}$  is computed.

### Mobility Upwinding

The mobility of the phases entering or leaving a finite volume is mainly determined by the relative permeability of the phases in the cells from which the flow originates. It is therefore common practice to use an upwinding scheme for interface mobilities, Peaceman [14, p. 143]. I approximate the aqueous saturation  $s_a(x, t)$  by a piecewise constant function, and denote its value within each finite volume  $\Omega_i$  by  $s_{a_i}(t)$ . The upwind mobility at the interface  $\gamma_{ij}$  is then defined as:

$$\lambda_{ij}(t) = \lambda(s_{a_i}(t)), \quad \text{if } p_i(t) > p_j(t), \quad (3.13a)$$

$$\lambda_{ij}(t) = \lambda(s_{a_j}(t)), \quad \text{if } p_i(t) < p_j(t), \quad (3.13b)$$

where it is understood that  $\lambda_{ij}(t)$  depends on both aqueous saturation and pressure at time  $t$ , but I simplify the notation. In the case of equal cell pressures  $p_i(t) = p_j(t)$ , we can choose either one of the two saturation values, or simply set  $\lambda_{ij}(t) = 0$ , since no flux is present. Using the upwind mobility and definition of  $\Delta p_{ij}$ , equation (3.12b)

becomes:

$$\sum_{j \in \mathcal{N}(i)} \int_{\gamma_{ij}} \lambda_{ij}(t) K_{ij} \frac{\Delta p_{ij}(t)}{h} dS = \int_{\Omega_i} q(x, t) dx. \quad (3.14)$$

The integrand on the left is independent of the integration variable and can be moved out. Setting:

$$\begin{aligned} F_{ij} &\stackrel{\text{def}}{=} \int_{\gamma_{ij}} dS, \\ q_i(t) &\stackrel{\text{def}}{=} \int_{\Omega_i} q(x, t) dx, \end{aligned}$$

yields the discretized pressure equation:

$$\sum_{j \in \mathcal{N}(i)} \lambda_{ij}(t) F_{ij} K_{ij} \frac{\Delta p_{ij}(t)}{h} = q_i(t). \quad (3.15)$$

It is common in the petroleum literature to define the *interface transmissibility*:

$$T_{ij} \stackrel{\text{def}}{=} F_{ij} K_{ij} / h, \quad (3.16)$$

and write (3.15) in the compact form

$$\sum_{j \in \mathcal{N}(i)} T_{ij} \lambda_{ij}(t) \Delta p_{ij}(t) = q_i(t). \quad (3.17)$$

### Computation of Interface Permeability $K_{ij}$

In equation (3.12b), I introduced the scalar interface permeability  $K_{ij}$ , which approximates the action of the permeability tensor  $K(x)$  on the pressure gradient along the interface  $\gamma_{ij}$ . In order to derive an expression for  $K_{ij}$ , I first approximate  $K(x)$  by a



piecewise constant function, where

$$K_i(x) \stackrel{\text{def}}{=} \begin{pmatrix} K_{i1}(x) & 0 \\ 0 & K_{i2}(x) \end{pmatrix}, \quad i = 1, \dots, N, \quad (3.18a)$$

denotes the average cell permeability for each finite volume  $\Omega_i$ . For the following derivation, it is sufficient to consider one dimensional flow, since I assume a uniform, rectangular, and  $k$ -orthogonal mesh for the finite volume discretization.

Consider the 1D diffusion equation:

$$-\left(k(x) u'(x)\right)' = f(x) \quad x \in (0, 1), \quad (3.19)$$

with homogeneous Neumann boundary conditions. Let  $\{x_i\}$  denote the grid points of a cell-centered mesh, and let  $x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}$  denote the location of the cell interfaces. Assume that  $k(x)$  is continuous within a cell, and let  $h > 0$  denote the uniform mesh-size. Now define the average diffusion coefficient:

$$k_i \stackrel{\text{def}}{=} \frac{1}{h} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} k(x) dx, \quad (3.20)$$

and compute two flux approximations for the interface between the cells  $i, i+1$ :

$$H_{i+\frac{1}{2}} = -k_i \frac{u_{i+\frac{1}{2}} - u_i}{h/2}, \quad (3.21a)$$

$$\tilde{H}_{i+\frac{1}{2}} = -k_{i+1} \frac{u_{i+1} - u_{i+\frac{1}{2}}}{h/2}. \quad (3.21b)$$

The continuity constraint for the interface flux dictates:  $H_{i+\frac{1}{2}} = \tilde{H}_{i+\frac{1}{2}}$ . Hence,

$$u_{i+\frac{1}{2}} = \frac{k_i u_i + k_{i+1} u_{i+1}}{k_i + k_{i+1}}. \quad (3.22)$$

Substituting the expression for  $u_{i+\frac{1}{2}}$  from (3.22) back into equation (3.21a), we obtain:

$$H_{i+\frac{1}{2}} = \frac{2}{h} \frac{k_i k_{i+1}}{k_i + k_{i+1}} (u_{i+1} - u_i). \quad (3.23)$$

Equation (3.23) suggests a harmonic averaging for the permeabilities of neighboring cells along their interface. Adapted to the numerical reservoir model, this leads to the following definition:

$$K_{ij} \stackrel{\text{def}}{=} \frac{K_{i1}(x)K_{j1}(x)}{K_{i1}(x) + K_{j1}(x)}, \quad \text{cells aligned along coordinate axis } x_1, \quad (3.24a)$$

$$K_{ij} \stackrel{\text{def}}{=} \frac{K_{i2}(x)K_{j2}(x)}{K_{i2}(x) + K_{j2}(x)}, \quad \text{cells aligned along coordinate axis } x_2. \quad (3.24b)$$

### Spatial Discretization of the Saturation Equation

The finite volume discretization of the saturation equation follows the same steps as the discretization of the pressure equation. Setting:

$$q_{a_i}(t) \stackrel{\text{def}}{=} \int_{\Omega_i} q_a(x, t) dx, \quad (3.25)$$

and defining the quantities:

$$V_i \stackrel{\text{def}}{=} \int_{\Omega_i} dx, \quad (3.26)$$

$$\phi_i \stackrel{\text{def}}{=} \text{average porosity for finite volume } \Omega_i, \quad (3.27)$$

I arrive at:

$$\frac{d}{dt} s_{a_i}(t) = \frac{1}{\phi_i V_i} \left( q_{a_i}(t) - \sum_{j \in \mathcal{N}(i)} T_{ij} \lambda_{a_{ij}}(t) \Delta p_{ij}(t) \right), \quad (3.28)$$

where  $\lambda_{a_{ij}}$  is the upwind aqueous phase mobility. For the nonlinear saturation equation, upwinding is essential for the stability of the finite volume discretization, see K.

W. Morton and T. Sonar [12]. To summarize the above, the finite volume discretization for the numerical reservoir model yields the following equations:

$$q_i(t) = \sum_{j \in \mathcal{N}(i)} T_{ij} \lambda_{ij}(t) \Delta p_{ij}(t), \quad (3.29a)$$

$$\frac{d}{dt} s_{a_i}(t) = \frac{1}{\phi_i V_i} \left( q_{a_i}(t) - \sum_{j \in \mathcal{N}(i)} T_{ij} \lambda_{a_{ij}}(t) \Delta p_{ij}(t) \right), \quad t > 0, \quad (3.29b)$$

$$s_{a_i}(0) = s_{a_i}^{init}, \quad (3.29c)$$

where  $\lambda_{ij}(t)$ ,  $\lambda_{a_{ij}}(t)$  are functions of the piecewise constant approximation of the aqueous saturations, and the piecewise linear approximation of the pressure field. Next, I introduce a compact matrix notation for the semi-discretized state equations.

### Matrix Formulation

The finite volume equations (3.29) can be assembled for the domain  $\Omega$  using a compact matrix notation. I first define the following matrix elements:

$$a_{ij}(t) = -T_{ij} \lambda_{ij}(t), \quad (3.30a)$$

$$a_{ii}(t) = \sum_{j \in \mathcal{N}(i)} T_{ij} \lambda_{ij}(t), \quad (3.30b)$$

$$\tilde{a}_{ij}(t) = -T_{ij} \lambda_{a_{ij}}(t), \quad (3.30c)$$

$$\tilde{a}_{ii}(t) = \sum_{j \in \mathcal{N}(i)} T_{ij} \lambda_{a_{ij}}(t). \quad (3.30d)$$

Remembering that  $\lambda_{ij}(t)$ ,  $\lambda_{a_{ij}}(t)$  depend on both saturations and pressures, we form the two matrices  $A(\vec{s}_a(t), \vec{p}(t))$  and  $\tilde{A}(\vec{s}_a(t), \vec{p}(t))$  using the elements defined above.

We now assemble equations (3.29a) for the finite volumes  $i = 1, \dots, N$  in the following

form:

$$\vec{q}(t) = A(\vec{s}_a(t), \vec{p}(t))\vec{p}(t). \quad (3.31)$$

Similarly, equations (3.29b) are assembled by:

$$\frac{d}{dt}\vec{s}_a(t) = D^{-1} \left( \vec{q}_a(t) - \tilde{A}(\vec{s}_a(t), \vec{p}(t))\vec{p}(t) \right), \quad (3.32)$$

where  $D^{-1}$  is a diagonal matrix with entries  $1/\phi_i V_i$ .

We finalize the step of writing the finite volume equations for the reservoir  $\Omega$  in compact form by defining two functions  $f, g$ :

$$g(t, \vec{s}_a(t), \vec{p}(t), \vec{q}(t)) \stackrel{\text{def}}{=} \vec{q}(t) - A(\vec{s}_a(t), \vec{p}(t))\vec{p}(t) = 0, \quad (3.33a)$$

$$f(t, \vec{s}_a(t), \vec{p}(t), \vec{q}(t)) \stackrel{\text{def}}{=} D^{-1} \left( \vec{q}_a(t) - \tilde{A}(\vec{s}_a(t), \vec{p}(t))\vec{p}(t) \right) = \frac{d}{dt}\vec{s}_a(t), \quad t > 0. \quad (3.33b)$$

Throughout the rest of the document, I drop the vector notation for  $\vec{s}_a(t)$ ,  $\vec{p}(t)$ ,  $\vec{q}(t)$ , and use subscripts to indicate that a particular element of the vector is being accessed.

The MATLAB implementation of the finite volume spatial discretization of the reservoir equations follows the implementation described by Aarnes, Gimse, and Lie [1].

## 3.6 Time Discretization

The two commonly used time discretization schemes in numerical reservoir simulation are IMPSAT and SEQUENTIAL. In the following I describe the two approaches in the context of my numerical reservoir model.

### 3.6.1 IMPSAT

The IMPSAT procedure simultaneously and implicitly solves for *end of time step* values of pressures and phase saturations, using a backward Euler time integration scheme which is known to be unconditionally stable, although large time steps will decrease accuracy due to linearization errors.

For my numerical reservoir model, I subdivide the simulation time interval  $[0, T]$  into  $K$  time steps of equal length. Let  $t^k$ ,  $s_a^k$ , and  $p^k$  denote end of time step values, while  $q^k$  denotes well rates held constant *during* the time step, that is, for  $t \in [(k-1)\Delta t, k\Delta t)$ . The IMPSAT algorithm is then given by:

**Algorithm 3.6.1** *IMPSAT- Algorithm*

*Load initial aqueous saturations  $s_a^0$*

*For  $k = 1, \dots, K$  do*

*Set well rates  $q^k$ .*

$$\text{Solve} \quad \begin{pmatrix} g(t^k, s_a^k, p^k, q^k) \\ s_a^k - s_a^{k-1} - \Delta t f(t, s_a^k, p^k, q^k) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{for } p^k, s_a^k.$$

*End*

### 3.6.2 SEQUENTIAL

The sequential time stepping procedure uses an operator splitting scheme. It differs from IMPSAT in the way that the two equations for pressure and saturations are solved sequentially, with the pressure equation being solved first. I first describe the differences between the sequential approach used in the industry and my approach, before outlining the algorithm used in my numerical reservoir simulator.

#### Sequential approach used in industrial simulators

The majority of industrial reservoir simulators implement the *volume balance* formulation, see Watts [24] for a reference. In this approach, the pressure equation is derived from a volume balance relationship, which states that the fluid volume must equal the pore volume at the end of the time step. This approach assumes compressible rock and fluid. Industrial simulators use the sequential formulation mainly to reduce computational time. They achieve this goal in the following way: First, the implicit pressure equation is solved using backward Euler time integration and a full Newton loop. Then, the solution to the implicit saturation equation is approximated using backward Euler with only a single Newton iteration. The resulting discrepancy in fluid and pore volume is used as a source term in the next time step, in order to correct the trajectory of the state variables.

### My SEQUENTIAL Approach

In my formulation, using incompressible rock and fluid, I use a different sequential scheme. After solving for new pressures using (3.33a), I solve (3.33b) for end of time step saturations using backward Euler time integration with a full Newton loop. I implement a dual time stepping scheme where pressures are computed based on larger time steps, and saturations are computed using a series of smaller time steps. The procedure works as follows.

I divide the simulation time interval  $[0, T]$  into  $K^p \in \mathbb{N}$  (pressure) time steps of equal length. Each of these time steps is subdivided into  $L \in \mathbb{N}$  smaller saturation time steps of equal size  $\Delta t$ . The total number of saturation time steps is then  $K^s = L \cdot K^p$ . I solve the discretized pressure equation at times  $kL\Delta t$  for  $k = 0, \dots, K^p - 1$ , and denote the solution by  $p^k \stackrel{\text{def}}{=} p(kL\Delta t)$ . The pressure is held constant over the interval  $t \in [kL\Delta t, (k+1)L\Delta t)$  when solving for saturations. I then compute saturations  $s_a^{kL+l} \stackrel{\text{def}}{=} s_a((kL+l)\Delta t)$ ,  $(l = 1, \dots, L)$  implicitly. Well rates are held constant during this time, since a change in rates would perturb the pressure field. In the following algorithm,  $p^k$ ,  $q^k$  denote the set of pressures and well rates in the half open interval  $[kL\Delta t, (k+1)L\Delta t)$ .

**Algorithm 3.6.2** *Sequential-Procedure*

*Load initial aqueous saturations  $s_a^0$*

*For  $k = 0, \dots, K^p - 1$  do*

*Set well rates  $q^k$*

*Solve  $g(t^{kL}, s_a^{kL}, p^k, q^k) = 0$  for  $p^k$*

*For  $l = 1, \dots, L$  do*

*Solve  $\frac{s_a^{kL+l} - s_a^{kL+l-1}}{\Delta t} = f(t^{kL+l}, s_a^{kL+l}, p^k, q^k)$  for  $s_a^{kL+l}$*

*End*

*End*

In the following chapter, I will use both formulations to derive derivatives, used in the solution of a reservoir optimization problem introduced in chapter 5.



## Chapter 4

# Optimal Well Rate Allocation

Based on the numerical reservoir model developed in the previous chapter, I want to compute injection and production well rates that optimize a given cost function over a fixed simulation period. I define the optimal control problem and then derive first and second order derivatives using the adjoint approach.

## 4.1 Optimal Control Problem

Let  $P, I$  denote the sets of finite volume indices for which the source terms:

$$q_i(t), i \in P \cup I, 1 \leq i \leq N,$$

are potentially nonzero, thus representing production and injection wells connected to the reservoir. Define

$$f_a(s_a) := \frac{\lambda_a}{\lambda_a + \lambda_l}(s_a), \quad (4.1a)$$

$$f_l(s_a) := \frac{\lambda_l}{\lambda_a + \lambda_l}(s_a), \quad (4.1b)$$

to be the fractional flow terms which determine the composition of produced fluids. Let  $q_{l_i}(t)$  denote the oil component of  $q_i(t)$ , and let  $q_{a_i}(t)$  denote the water component of  $q_i(t)$  for a production well. We compute  $q_{l_i}(t)$ ,  $q_{a_i}(t)$  as follows:

$$q_{l_i}(s_{a_i}(t)) = q_i(t) f_l(s_{a_i}(t)), \quad (4.2a)$$

$$q_{a_i}(s_{a_i}(t)) = q_i(t) f_a(s_{a_i}(t)). \quad (4.2b)$$

For injection wells, we set  $f_a = 1$  and  $f_l = 0$  for all  $t$ , and hence, the source terms do not depend on the saturation values. Given these definitions, and using the convention that production rates are negative, and injection rates are positive, I pose the following semi-discretized optimal control problem, where I want to find well-rates

$q(t)$  to minimize a given cost function:

$$\min_q \int_0^T \sum_{i \in P} \alpha q_{l_i}(s_{a_i}(t)) + \sum_{i \in P} \gamma q_{a_i}(s_{a_i}(t)) + \sum_{i \in I} \frac{\beta}{2} q_i^2(t) dt, \quad (4.3a)$$

subject to:

$$e^T q(t) = 0, \quad t \in [0, T], \quad (4.3b)$$

$$q_{\min} \leq q_i(t) \leq q_{\max}, \quad t \in [0, T], \quad (4.3c)$$

where  $p(t), s_a(t), q(t)$  solve the finite volume based state equations:

$$g(t, s_a(t), p(t), q(t)) = 0, \quad (4.3d)$$

$$f(t, s_a(t), p(t), q(t)) = \frac{d}{dt} s_a(t), \quad t > 0, \quad (4.3e)$$

and where  $\alpha$  is a factor for determining revenue from oil production, and  $\beta$  and  $\gamma$  are factors that penalize water injection and production. The equality constraint (4.3b) is required by the Neumann boundary condition, imposed on the system due to incompressible fluid and rock.

Problem 4.3 is an implicitly constraint optimal control problem, in which the state equations form the implicit constraints for the state and control variables. Formulating the problem in this way allows me to maintain a higher degree of separation between the implementation of the optimization problem and the numerical reservoir simulator.

## 4.2 Adjoint and Gradient Computations

I first describe the gradient computation for an implicitly constrained problem using an abstract example, and then derive the equations for the optimal control problem. The derivations for the abstract example problem have been provided by my advisor Dr. M. Heinkenschloss [8] and are used herein with his permission.

### 4.2.1 Gradient Computation for an Abstract Problem

Consider:

$$\min_{u \in U} \hat{\Psi}(u) = \Psi(y(u), u), \quad (4.4a)$$

where  $y(u) \in R^{n_y}$  is the solution of the nonlinear state equation

$$c(y, u) = 0, \quad (4.4b)$$

and where

$$\Psi : R^{n_y+n_u} \rightarrow R, \quad c : R^{n_y+n_u} \rightarrow R^{n_y}. \quad (4.4c)$$

#### Assumption 4.2.1

*For all  $u \in U$ , there exists exactly one  $y \in R^{n_y}$  such that  $c(y, u) = 0$ .*

*There exists an open set  $D \subset R^{n_y+n_u}$  with  $\{(y, u) : u \in U, c(y, u) = 0\} \subset D$ ,*

*such that  $\Psi, c$  are twice continuously differentiable on  $D$ .*

*The inverse  $\left(\frac{\partial}{\partial y} c(y, u)\right)^{-1}$  exists for all  $(y, u) \in \{(y, u) : u \in U, c(y, u) = 0\}$ .*

Under assumptions 4.2.1, the implicit function theorem states the existence of a differentiable function  $y : R^{n_u} \rightarrow R^{n_y}$ , defined by  $c(y(u), u) = 0$ .

### Gradient Computation using Sensitivities

The implicit function theorem implies the differentiability of  $\Psi$ , with the Jacobian of  $\Psi$  being the solution of:

$$c_y(y, u)|_{y=y(u)} \cdot y_u(u) = -c_u(y, u)|_{y=y(u)}. \quad (4.5)$$

To simplify notation, I use  $c_y(y(u), u)$  for  $c_y(y, u)|_{y=y(u)}$  and make similar simplifications in other terms where the meaning is obvious. Using this notation, we have:

$$y_u(u) = -c_y(y(u), u)^{-1} c_u(y(u), u). \quad (4.6)$$

The derivative  $y_u(u)$  is called the *sensitivity* of  $y$  with respect to the control variable  $u$ . Since  $y(u)$  is differentiable,  $\hat{\Psi}$  is differentiable and its gradient is given by

$$\nabla \hat{\Psi}(u) = y_u(u)^T \nabla_y \Psi(y(u), u) + \nabla_u \Psi(y(u), u) \quad (4.7a)$$

$$= -c_u(y(u), u)^T c_y(y(u), u)^{-T} \nabla_y \Psi(y(u), u) + \nabla_u \Psi(y(u), u). \quad (4.7b)$$

### Gradient Computation using Adjoint

If in (4.7), we define  $\lambda(u) := -c_y(y(u), u)^{-T} \nabla_y \Psi(y(u), u)$ , the gradient of  $\hat{\Psi}$  can be written as:

$$\nabla \hat{\Psi}(u) = \nabla_u \Psi(y(u), u) + c_u(y(u), u)^T \lambda(u). \quad (4.8a)$$

Here,  $\lambda(u)$  is the solution of the *Adjoint Equation*:

$$c_y(y(u), u)^T \lambda = -\nabla_y \Psi(y(u), u). \quad (4.8b)$$

The above suggests the following algorithm for computing the gradient of  $\hat{\Psi}$ :

Given  $u$ , solve the state equation  $c(y, u) = 0$  for  $y$ .

Solve the adjoint equation  $c_y(y(u), u)^T \lambda = -\nabla_y \Psi(y(u), u)$  for  $\lambda$ .

Denote the solution by  $\lambda(u)$ .

Compute the gradient  $\nabla \hat{\Psi}(u) = \nabla_u \Psi(y(u), u) + c_u(y(u), u)^T \lambda(u)$ .

### Lagrangian

The gradient computation using the adjoint method can also be expressed by defining the Lagrangian:

$$L(y, u, \lambda) := \Psi(y, u) + \lambda^T \cdot c(y, u), \quad (4.9a)$$

where the *Lagrange Multiplier*  $\lambda$  relaxes the implicit constraint (4.4b).

Using the Lagrangian, equation (4.8b) can be written as:

$$\nabla_y L(y, u, \lambda)|_{y=y(u), \lambda=\lambda(u)} = 0. \quad (4.9b)$$

Moreover, equation (4.8a) can then be written as:

$$\nabla \hat{\Psi}(u) = \nabla_u L(y, u, \lambda) \Big|_{y=y(u), \lambda=\lambda(u)}. \quad (4.9c)$$

### Reason for Using the Adjoint Approach

Examination of the sensitivity term  $y_u(u) = -c_y(y(u), u)^{-1} c_u(y(u), u)$  shows that it requires the solution of  $n_u$  systems of linear equations. The approach becomes especially expensive if the control space is large. However, the gradient computation (4.7)

requires only the application of the *transpose* of  $y_u(u)$  to  $\nabla_y \Psi(y(u), u)$ . Therefore, the adjoint approach is more efficient. The use of the Lagrangian is an elegant way to derive the adjoint.

### 4.2.2 Gradient Computation for Semi-Discretized Case

To simplify notation, I define the cost function:

$$l(t, s_a(t), q(t)) \stackrel{\text{def}}{=} \sum_{i \in P} \alpha q_{l_i}(s_{a_i}(t)) + \sum_{i \in P} \gamma q_{a_i}(s_{a_i}(t)) + \sum_{i \in I} \frac{\beta}{2} q_i^2(t), \quad (4.10)$$

where  $q_{l_i}(s_{a_i}(t))$ ,  $q_{a_i}(s_{a_i}(t))$  are as defined in (4.2). The Lagrangian for the semi-discretized optimization problem (4.3) is then given by:

$$\begin{aligned} L(p, s_a, q, \lambda_p, \lambda_s) &= \int_0^T l(t, s_a(t), q(t)) dt \\ &+ \int_0^T \lambda_s(t)^T [s'_a(t) - f(t, s_a(t), p(t), q(t))] dt \\ &+ \int_0^T \lambda_p(t)^T g(t, s_a(t), p(t), q(t)) dt. \end{aligned} \quad (4.11)$$

The gradient of the objective function is given by:

$$\nabla J(q) = \nabla_q L(p, s_a, q, \lambda_p, \lambda_s), \quad (4.12)$$

where  $p = p(q)$ ,  $s_a = s_a(q)$  are the solution of the dynamic state equations, and

$\lambda_p = \lambda_p(q)$ ,  $\lambda_s = \lambda_s(q)$  are the solution of the adjoint equation:

$$\frac{\partial}{\partial s_a} L(p, s_a, q, \lambda_p, \lambda_s) = 0, \quad (4.13a)$$

$$\frac{\partial}{\partial p} L(p, s_a, q, \lambda_p, \lambda_s) = 0. \quad (4.13b)$$

To derive the adjoint differential equation, we compute

$$\begin{aligned}
\frac{\partial}{\partial s_a} L(p, s_a, q, \lambda_p, \lambda_s) \delta s_a &= \int_0^T \nabla_s l(t, s_a(t), q(t))^T \delta s_a(t) dt \\
&\quad + \int_0^T \lambda_s(t)^T [\delta s'_a(t) - \frac{\partial}{\partial s_a} f(t, s_a(t), p(t), q(t)) \delta s_a(t)] dt \\
&\quad + \int_0^T \lambda_p(t)^T \frac{\partial}{\partial s_a} g(t, s_a(t), p(t), q(t)) \delta s_a(t) dt, \\
\frac{\partial}{\partial p} L(p, s_a, q, \lambda_p, \lambda_s) \delta p &= - \int_0^T \lambda_s(t)^T \frac{\partial}{\partial p} f(t, s_a(t), p(t), q(t)) \delta p(t) dt \\
&\quad + \int_0^T \lambda_p(t)^T \frac{\partial}{\partial p} g(t, s_a(t), p(t), q(t)) \delta p(t) dt.
\end{aligned}$$

Integrating by parts, and setting both derivatives to zero yields the semi-discrete adjoint equations:

$$\begin{aligned}
0 &= \frac{\partial}{\partial p} g(t, s_a(t), p(t), q(t))^T \lambda_p(t) \\
&\quad - \frac{\partial}{\partial p} f(t, s_a(t), p(t), q(t))^T \lambda_s(t), \quad t \in (0, T) \quad (4.14a)
\end{aligned}$$

$$\begin{aligned}
-\frac{d}{dt} \lambda_s(t) &= \frac{\partial}{\partial s_a} f(t, s_a(t), p(t), q(t))^T \lambda_s(t) \\
&\quad - \frac{\partial}{\partial s_a} g(t, s_a(t), p(t), q(t))^T \lambda_p(t) \\
&\quad - \nabla_s l(t, s_a(t), q(t)), \quad (4.14b)
\end{aligned}$$

$$\lambda_s(T) = 0. \quad (4.14c)$$

The gradient (4.12) is then given in the context of the linearization:

$$DJ(q) \delta q = \int_0^T \left[ \underbrace{\nabla_q l(t, s_a, q) - \lambda_s(t)^T D_q f(t, s_a, p, q) + \lambda_p(t)^T D_q g(t, s_a, p, q)}_{\nabla J(q)} \right]^T \delta q(t) dt.$$



### Testing the Adjoint Computation

In the following text, I will use the notation  $D_{s_a}f$ ,  $D_pg$ , etc. to indicate derivatives.

If  $\lambda_p, \lambda_s$  solve (4.14) and if  $\tilde{p}, \tilde{s}_a$  solve the linearized state equation:

$$\begin{aligned}\tilde{s}_a'(t) &= D_{s_a}f(t, s_a(t), p(t), q(t))\tilde{s}_a(t) \\ &\quad + D_pf(t, s_a(t), p(t), q(t))\tilde{p}(t) + r_{s_a}(t), \quad t \in (0, T)\end{aligned}\tag{4.15a}$$

$$\begin{aligned}0 &= D_{s_a}g(t, s_a(t), p(t), q(t))\tilde{s}_a(t) \\ &\quad + D_pg(t, s_a(t), p(t), q(t))\tilde{p}(t) + r_p(t), \quad t \in (0, T)\end{aligned}\tag{4.15b}$$

$$\tilde{s}_a(0) = 0,\tag{4.15c}$$

where  $r_{s_a}, r_p$  are arbitrary functions, then multiplying (4.14a,b) by  $\tilde{s}_a, \tilde{p}$ , respectively, integrating the resulting identities over  $[0, T]$ , and applying integration by parts leads to:

$$\begin{aligned}0 &= \int_0^T \nabla_{s_a}l(t, s_a(t), q(t))^T \tilde{s}_a(t) dt \\ &\quad + \int_0^T \lambda_s(t)^T [\tilde{s}_a'(t) - D_{s_a}f(t, s_a(t), p(t), q(t))\tilde{s}_a(t)] dt \\ &\quad + \int_0^T \lambda_p(t)^T D_{s_a}g(t, s_a(t), p(t), q(t))\tilde{s}_a(t) dt,\end{aligned}\tag{4.16a}$$

$$\begin{aligned}0 &= - \int_0^T \lambda_s(t)^T D_pf(t, s_a(t), p(t), q(t))\tilde{p}(t) dt \\ &\quad + \int_0^T \lambda_p(t)^T D_pg(t, s_a(t), p(t), q(t))\tilde{p}(t) dt.\end{aligned}\tag{4.16b}$$

Adding both equations and using (4.15) gives:

$$0 = \int_0^T \nabla_{s_a}l(t, s_a(t), q(t))^T \tilde{s}_a(t) dt + \int_0^T \lambda_s(t)^T r_{s_a}(t) dt - \int_0^T \lambda_p(t)^T r_p(t) dt. \tag{4.16c}$$

The latter identity must be satisfied for all functions  $r_{s_a}, r_p$ . Hence, if a solver for the linearized state equation (4.15) is available, then the solution of the adjoint equations (4.14) must satisfy (4.16c).

### 4.2.3 Gradient Computation IMPSAT

For IMPSAT, we discretize the objective function (4.3a) in the following way:

$$J(q) = \Delta t \sum_{k=1}^K l(t^k, s_a^k, q^k). \quad (4.17)$$

The optimal control problem (4.3) becomes:

$$\text{Minimize } \Delta t \sum_{k=1}^K l(t^k, s_a^k, q^k), \quad (4.18a)$$

$$\text{subject to } \frac{s_a^k - s_a^{k-1}}{\Delta t} = f(t^k, s_a^k, p^k, q^k), \quad k = 1, \dots, K \quad (4.18b)$$

$$0 = g(t^k, s_a^k, p^k, q^k) \quad k = 1, \dots, K. \quad (4.18c)$$

The Lagrangian corresponding to optimization model problem (4.18) is given by:

$$\begin{aligned} L(p, s_a, q, \lambda_p, \lambda_s) = & \Delta t \sum_{k=1}^K l(t^k, s_a^k, q^k) + \sum_{k=1}^K [\lambda_s^k]^T [s_a^k - s_a^{k-1} - \Delta t f(t^k, s_a^k, p^k, q^k)] \\ & + \sum_{k=1}^K \Delta t [\lambda_p^k]^T g(t^k, s_a^k, p^k, q^k). \end{aligned} \quad (4.19)$$

### IMPSAT Adjoint Equations

Setting the partial derivatives of the Lagrangian with respect to  $s_a^k$  and  $p^k$ ,  $k = 1, \dots, K$ , to zero gives the discrete adjoint equations. I set  $\lambda_s^{K+1} = 0$  in accordance with (4.14c). The adjoint is computed as follows:

**Algorithm 4.2.2 (Adjoint Computation IMPSAT)**

For  $k = K, \dots, 1$  solve

$$-\frac{\lambda_s^{k+1} - \lambda_s^k}{\Delta t} = D_s f(t^k, s_a^k, p^k, q^k)^T \lambda_s^k - D_s g(t^k, s_a^k, p^k, q^k)^T \lambda_p^k - \nabla_{s_a} l(t^k, s_a^k, q^k), \quad (4.20a)$$

$$0 = D_p g(t^k, s_a^k, p^k, q^k)^T \lambda_p^k - D_p f(t^k, s_a^k, p^k, q^k)^T \lambda_s^k. \quad (4.20b)$$

Since we have assumed that  $D_p g(t, s, p, q)$  is invertible, (4.20) can be solved for  $\lambda_p^K$ , which yields:

$$\begin{aligned} \lambda_s^K &= \Delta t \left[ D_s f(t^K, s_a^K, p^K, q^K)^T \right. \\ &\quad \left. - D_s g(t^K, s_a^K, p^K, q^K)^T [D_p g(t^K, s_a^K, p^K, q^K)]^{-T} D_p f(t^K, s_a^K, p^K, q^K)^T \right] \lambda_s^K \\ &\quad - \Delta t \nabla_{s_a} l(t^K, s_a^K, q^K). \end{aligned} \quad (4.21)$$

For sufficiently small  $\Delta t$ , (4.21) has a unique solution  $\lambda_s^K$  which satisfies  $\|\lambda_s^K\| = O(\Delta t)$ . Asymptotically, this matches the final condition (4.14c).

**Testing the Adjoint Computation**

Let  $\lambda_p^k, \lambda_s^k$ ,  $k = 1, \dots, K$  solve (4.20). Multiplying (4.20a) by  $\tilde{s}_a^k$ ,  $k = 1, \dots, K$ , and multiplying (4.20b) by  $\tilde{p}^k$ ,  $k = 1, \dots, K$ , setting  $\tilde{s}_a^0 = 0$ , and summing the resulting

equations gives:

$$\begin{aligned}
0 &= \sum_{k=1}^K \Delta t \nabla_{s_a} l(t^k, s_a^k, q^k) \tilde{s}_a^k \\
&+ \sum_{k=1}^K [\lambda_s^k]^T [\tilde{s}_a^k - \tilde{s}_a^{k-1} - \Delta t D_{s_a} f(t^k, s_a^k, p^k, q^k) \tilde{s}_a^k - \Delta t D_p f(t^k, s_a^k, p^k, q^k) \tilde{p}^k] \\
&+ \sum_{k=1}^K \Delta t [\lambda_p^k]^T [D_{s_a} g(t^k, s_a^k, p^k, q^k) \tilde{s}_a^k + D_p g(t^k, s_a^k, p^k, q^k) \tilde{p}^k]. \tag{4.22a}
\end{aligned}$$

If we set  $\tilde{s}_a^0 = 0$ , and if  $\tilde{p}^k, \tilde{s}_a^k, k = 1, \dots, K$  solve the linearized state equation:

$$\frac{\tilde{s}_a^k - \tilde{s}_a^{k-1}}{\Delta t} = D_{s_a} f(t^k, s_a^k, p^k, q^k) \tilde{s}_a^k + D_p f(t^k, s_a^k, p^k, q^k) \tilde{p}^k + r_{s_a}^k, \tag{4.23a}$$

$$0 = D_{s_a} g(t^k, s_a^k, p^k, q^k) \tilde{s}_a^k + D_p g(t^k, s_a^k, p^k, q^k) \tilde{p}^k + r_p^k, \tag{4.23b}$$

then (4.22a) leads to:

$$0 = \sum_{k=1}^K \Delta t \nabla_{s_a} l(t^k, s_a^k, q^k) \tilde{s}_a^k + \sum_{k=1}^K \Delta t [\lambda_s^k]^T r_{s_a}^k - \sum_{k=1}^K \Delta t [\lambda_p^k]^T r_p^k. \tag{4.24}$$

The identity (4.24) has to hold for all sequences  $\{\tilde{s}_a^k\}, \{\tilde{p}^k\}$ .

## IMPSAT Gradient Computation

For  $k = 1, \dots, K$ , the IMPSAT gradient is computed as:

$$\begin{aligned}
\nabla_{q^k} J(q) &= \nabla_{q^k} l(t^k, s_a^k, q^k) - \left[ D_q f(t^k, s_a^k, p^k, q^k) \right]^T \lambda_s^k \\
&+ \left[ D_q g(t^k, s_a^k, p^k, q^k) \right]^T \lambda_p^k. \tag{4.25}
\end{aligned}$$

#### 4.2.4 Gradient Computation SEQUENTIAL

For SEQUENTIAL, we discretize the objective function (4.3a) in the following way:

$$J(q) = \Delta t \sum_{k=0}^{K^p-1} \sum_{l=1}^L l(t^{kL+l}, s_a^{kL+l}, q^k), \quad (4.26)$$

where the meaning of  $L$  and  $K^p$  as well as the other quantities are as defined in section 3.6.2. The optimal control problem (4.3) becomes:

$$\min \Delta t \sum_{k=0}^{K^p-1} \sum_{l=1}^L l(t^{kL+l}, s_a^{kL+l}, q^k), \quad (4.27a)$$

subject to

$$0 = g(t^{kL}, s_a^{kL}, p^k, q^k), \quad (4.27b)$$

$$\frac{s_a^{kL+l} - s_a^{kL+l-1}}{\Delta t} = f(t^{kL+l}, s_a^{kL+l}, p^k, q^k), \quad (4.27c)$$

where  $k = 0, \dots, K^p - 1$  and  $l = 1, \dots, L$ .

We now denote  $\lambda_p(t^{kL})$  by  $\lambda_p^k$ . The Lagrangian is then given by:

$$\begin{aligned} L(p, s_a, q, \lambda_p, \lambda_s) = & \sum_{k=0}^{K^p-1} \left\{ \Delta t \sum_{l=1}^L l(t^{kL+l}, s_a^{kL+l}, q^k) \right. \\ & + L \Delta t [\lambda_p^k]^T g(t^{kL}, s_a^{kL}, p^k, q^k) \\ & \left. + \sum_{l=1}^L [\lambda_s^{kL+l}]^T \left[ s_a^{kL+l} - s_a^{kL+l-1} - \Delta t f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \right] \right\}, \quad (4.28) \end{aligned}$$

where I have weighted the discrete Lagrange multipliers with  $\Delta t$  and  $L \Delta t$  respectively.

## SEQUENTIAL Adjoint Equations

The partial derivatives of the Lagrangian with respect to pressures are given by:

$$\begin{aligned}\nabla_{p^k} L(p, s_a, q, \lambda_p, \lambda_s) &= L\Delta t \left[ D_p g(t^{kL}, s_a^{kL}, p^k, q^k) \right]^T \lambda_p^k \\ &\quad - \Delta t \sum_{l=1}^L \left[ D_p f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \right]^T \lambda_s^{kL+l}.\end{aligned}$$

The partial derivatives of the Lagrangian with respect to the aqueous saturations are calculated as follows.

For  $k = 0, \dots, K^p - 1$ ,  $l = 1, \dots, L - 1$ , compute:

$$\begin{aligned}\nabla_{s_a^{kL+l}} L(p, s_a, q, \lambda_p, \lambda_s) &= \Delta t \nabla_{s_a} l(t^{kL+l}, s_a^{kL+l}, q^k) \\ &\quad + \lambda_s^{kL+l} - \lambda_s^{kL+l+1} - \Delta t \left[ D_{s_a} f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \right]^T \lambda_s^{kL+l}.\end{aligned}$$

For  $k = 0, \dots, K^p - 2$ ,  $l = L$ , compute:

$$\begin{aligned}\nabla_{s_a^{kL+L}} L(p, s_a, q, \lambda_p, \lambda_s) &= \Delta t \nabla_{s_a} l(t^{kL+L}, s_a^{kL+L}, q^k) \\ &\quad + \lambda_s^{kL+L} - \lambda_s^{kL+L+1} - \Delta t \left[ D_{s_a} f(t^{kL+L}, s_a^{kL+L}, p^k, q^k) \right]^T \lambda_s^{kL+L} \\ &\quad + L\Delta t \left[ D_{s_a} g(t^{kL+L}, s_a^{kL+L}, p^{k+1}, q^{k+1}) \right]^T \lambda_p^{k+1}.\end{aligned}$$

For  $k = K^p - 1$ ,  $l = L$ , compute:

$$\begin{aligned}\nabla_{s_a^{kL+L}} L(p, s_a, q, \lambda_p, \lambda_s) &= \Delta t \nabla_{s_a} l(t^{kL+L}, s_a^{kL+L}, q^k) \\ &\quad + \lambda_s^{kL+L} - \Delta t \left[ D_{s_a} f(t^{kL+L}, s_a^{kL+L}, p^k, q^k) \right]^T \lambda_s^{kL+L}.\end{aligned}$$

The adjoint equations are obtained by setting the partial derivatives to zero. This results in the following procedure for the computations of adjoints.

**Algorithm 4.2.3 (Adjoint Computation SEQUENTIAL)**

Let  $K^s, K^p$  and  $L$  be given as in Algorithm 3.6.2. Let  $p^k, s_a^{kL+l}, k = 0, \dots, K^p - 1, l = 1, \dots, L$ , be solutions of the dynamic state equations.

For  $k = K^p - 1, \dots, 0$  do:

1. If  $k = K^p - 1$ , solve for  $\lambda_s^{kL+L}$

$$\lambda_s^{kL+L} = \Delta t \left[ D_{s_a} f(t^{kL+L}, s_a^{kL+L}, p^k, q^k) \right]^T \lambda_s^{kL+L} \quad (4.29a)$$

$$- \Delta t \nabla_{s_a} l(t^{kL+L}, s_a^{kL+L}, q^k). \quad (4.29b)$$

If  $k_p < K^p - 1$ , solve for  $\lambda_s^{kL+L}$

$$-\frac{\lambda_s^{kL+L+1} - \lambda_s^{kL+L}}{\Delta t} = \left[ D_{s_a} f(t^{kL+L}, s_a^{kL+L}, p^k, q^k) \right]^T \lambda_s^{kL+L} \quad (4.29c)$$

$$- L \left[ D_{s_a} g(t^{kL+L}, s_a^{kL+L}, p^{k+1}, q^{k+1}) \right]^T \lambda_p^{k+1} \quad (4.29d)$$

$$- \nabla_{s_a} l(t^{kL+L}, s_a^{kL+L}, q^k). \quad (4.29e)$$

2. For  $l = L - 1, \dots, 1$ , solve for  $\lambda_s^{kL+l}$ :

$$-\frac{\lambda_s^{kL+l+1} - \lambda_s^{kL+l}}{\Delta t} = \left[ D_{s_a} f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \right]^T \lambda_s^{kL+l} \quad (4.29f)$$

$$- \nabla_{s_a} l(t^{kL+l}, s_a^{kL+l}, q^k). \quad (4.29g)$$

3. Solve for  $\lambda_p^k$

$$\left[ D_p g(t^{kL}, s_a^{kL}, p^k, q^k) \right]^T \lambda_p^k = \sum_{l=1}^L \frac{1}{L} \left[ D_p f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \right]^T \lambda_s^{kL+l}. \quad (4.29h)$$

End ('k')

### Testing the Adjoint Computation

If  $\lambda_p^k, \lambda_s^{kL+l}$  solve the adjoint equations (4.29) for  $k = 0, \dots, K^p - 1, l = 1, \dots, L$ ,

then:

$$\begin{aligned}
0 &= \nabla_{s_a} L(p, s_a, q, \lambda_p, \lambda_s) \tilde{s}_a + \nabla_p L(p, s_a, q, \lambda_p, \lambda_s) \tilde{p} \\
&= \sum_{k=0}^{K^p-1} \left\{ \Delta t \sum_{l=1}^L \nabla_{s_a} l(t^{kL+l}, s_a^{kL+l}, q^k)^T \tilde{s}_a^{kL+l} \right. \\
&\quad + L \Delta t [\lambda_p^k]^T D_p g(t^{kL}, s_a^{kL}, p^k, q^k) \tilde{p}^k \\
&\quad + \sum_{l=1}^L [\lambda_s^{kL+l}]^T \left[ \tilde{s}_a^{kL+l} - \tilde{s}_a^{kL+l-1} - \Delta t D_{s_a} f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \tilde{s}_a^{kL+l} \right] \\
&\quad \left. - \sum_{l=1}^L \Delta t [\lambda_s^{kL+l}]^T D_p f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \tilde{p}^k \right\} \\
&\quad + \sum_{k=0}^{K^p-2} L \Delta t [\lambda_p^{k+1}]^T D_{s_a} g(t^{(k+1)L}, s_a^{(k+1)L}, p^{k+1}, q^{k+1}) \tilde{s}_a^{(k+1)L}. \tag{4.30a}
\end{aligned}$$

Setting  $\tilde{s}_a^0 = 0$ , equation (4.30a) can be re-written as:

$$\begin{aligned}
0 &= \sum_{k=0}^{K^p-1} \left\{ \Delta t \sum_{l=1}^L \nabla_{s_a} l(t^{kL+l}, s_a^{kL+l}, q^k)^T \tilde{s}_a^{kL+l} \right. \\
&\quad + L \Delta t [\lambda_p^k]^T D_{s_a} g(t^{kL}, s_a^{kL}, p^k, q^k) \tilde{s}_a^{kL} + L \Delta t [\lambda_p^k]^T D_p g(t^{kL}, s_a^{kL}, p^k, q^k) \tilde{p}^k \\
&\quad + \sum_{l=1}^L [\lambda_s^{kL+l}]^T \left[ \tilde{s}_a^{kL+l} - \tilde{s}_a^{kL+l-1} - \Delta t D_{s_a} f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \tilde{s}_a^{kL+l} \right] \\
&\quad \left. - \sum_{l=1}^L \Delta t [\lambda_s^{kL+l}]^T D_p f(t^{kL+l}, s_a^{kL+l}, p^k, q^k) \tilde{p}^k \right\} \tag{4.30b}
\end{aligned}$$

for all  $\tilde{s}_a = \left( (\tilde{s}_a^1)^T, \dots, (\tilde{s}_a^{K^s})^T \right)^T$  and all  $\tilde{p} = \left( (\tilde{p}^0)^T, \dots, (\tilde{p}^{K^p})^T \right)^T$ .



Hence, if  $\tilde{s}_a^0 = 0$ ,  $\lambda_p^{K^p} = 0$ , and if  $\tilde{s}_a^1, \dots, \tilde{s}_a^{K^s}$ ,  $\tilde{p}^0, \dots, \tilde{p}^{K^p}$  solve the linearized state equations:

$$0 = D_{s_a}g(t^{kL}, s_a^{kL}, p^k, q^k)\tilde{s}_a^{kL} + D_pg(t^{kL}, s_a^{kL}, p^k, q^k)\tilde{p}^k + r_p^k, \quad (4.31a)$$

$$\begin{aligned} \frac{\tilde{s}_a^{kL+l} - \tilde{s}_a^{kL+l-1}}{\Delta t} &= D_{s_a}f(t^{kL+l}, s_a^{kL+l}, p^k, q^k)\tilde{s}_a^{kL+l} \\ &+ D_pf(t^{kL+l}, s_a^{kL+l}, p^k, q^k)\tilde{p}^k + r_{s_a}^{kL+l}, \end{aligned} \quad (4.31b)$$

for arbitrary right hand side vectors  $r_{s_a}^{kL+l}, r_p^k$ ,  $k = 0, \dots, K^p - 1$ ,  $l = 1, \dots, L$ , then

$$\begin{aligned} 0 &= \sum_{k=0}^{K^p-1} \Delta t \left\{ \sum_{l=1}^L \nabla_{s_a} l(t^{kL+l}, s_a^{kL+l}, q^k)^T \tilde{s}_a^{kL+l} - [L\lambda_p^k]^T r_p^k \right. \\ &\quad \left. + \sum_{l=1}^L [\lambda_s^{kL+l}]^T r_{s_a}^{kL+l} \right\}. \end{aligned} \quad (4.31c)$$

### SEQUENTIAL Gradient Computation

Let  $K^s, K^p$  and  $L$  be given as in Algorithm 3.6.2. Let  $p^k, s_a^{kL+l}$ ,  $k = 0, \dots, K^p - 1$ ,  $l = 1, \dots, L$ , be solutions of the dynamic state equations, and let  $\lambda_p^k, \lambda_s^{kL+l}$  be the solution of the adjoint equations (4.29). The gradient is then computed as follows.

#### Algorithm 4.2.4 (Gradient Computation SEQUENTIAL)

For  $k = 0, \dots, K^p - 1$  compute:

$$\begin{aligned} \nabla_{q^k} J(q) &= \frac{1}{L} \sum_{l=1}^L \nabla_q l(t^{kL+l}, s_a^{kL+l}, q^k) + [D_q g(t^{kL}, s_a^{kL}, p^k, q^k)]^T \lambda_p^k \\ &\quad - \frac{1}{L} \sum_{l=1}^L [D_q f(t^{kL+l}, s_a^{kL+l}, p^k, q^k)]^T \lambda_s^{kL+l}, \end{aligned} \quad (4.32)$$

where the computation of the gradient is based on the inner product

$$\langle q_1, q_2 \rangle \equiv \sum_{k=0}^{K^p-1} L \Delta t (q_1^k)^T (q_2^k).$$

### 4.3 Second Order Derivatives

In this section I derive the algorithm to compute Hessian times Vector products for the fully discretized reservoir optimization problem, using the IMPSAT formulation.

I start by describing the derivation using the abstract model problem.

#### 4.3.1 Second Order Derivatives for Abstract Problem

##### Hessian Computation

Assuming that  $\Psi$  and  $c$  are twice continuously differentiable,  $\hat{\Psi}$  is twice continuously differentiable, and we can compute the Hessian from (4.9c). We first need the derivative  $\frac{d}{du}\lambda(u)$ . Applying the implicit function theorem to (4.9b) gives us:

$$\nabla_{yy}L(y(u), u, \lambda(u)) y_u(u) + \nabla_{yu}L(y(u), u, \lambda(u)) + \nabla_{y\lambda}L(y(u), u, \lambda(u)) \lambda(u) = 0.$$

Using the fact that  $\nabla_{y,\lambda}L(y, u, \lambda) = c_y(y, u)^T$  and making use of (4.6) in the previous equation, we find that

$$\begin{aligned} \lambda_u(u) = & c_y(y(u), u)^{-T} [\nabla_{yy}L(y(u), u, \lambda(u)) c_y(y(u), u)^{-1} c_u(y(u), u) \\ & - \nabla_{yu}L(y(u), u, \lambda(u))]. \end{aligned} \quad (4.33a)$$

We can now differentiate (4.9c):

$$\begin{aligned} \nabla^2 \hat{\Psi}(u) = & \nabla_{uy}L(y(u), u, \lambda(u)) y_u(u) + \nabla_u^2 L(y(u), u, \lambda(u)) \\ & + \nabla_{u,\lambda}L(y(u), u, \lambda(u)) \lambda_u(u). \end{aligned} \quad (4.33b)$$

If we insert (4.33a) and (4.6) in (4.33b) and observe that  $\nabla_{u\lambda}L(y(u), u, \lambda(u)) = c_u(y(u), u)$ , the Hessian can be written as:

$$\begin{aligned}\nabla^2\hat{\Psi}(u) &= c_u(y(u), u)^T c_y(y(u), u)^{-T} \nabla_y^2 L(y(u), u, \lambda(u)) c_y(y(u), u)^{-1} c_u(y(u), u) \\ &\quad - c_u(y(u), u)^T c_y(y(u), u)^{-T} \nabla_{yu} L(y(u), u, \lambda(u)) \\ &\quad - \nabla_{uy} L(y(u), u, \lambda(u)) c_y(y(u), u)^{-1} c_u(y(u), u) + \nabla_u^2 L(y(u), u, \lambda(u)).\end{aligned}\tag{4.33c}$$

### Hessian times Vector Computation

The computation and storage of the full Hessian is often infeasible unless sparsity can be exploited. A way out is to use optimization algorithms that rely on Hessian-times-vector products, which can be computed as follows.

#### Algorithm 4.3.1 Hessian times Vector Computation - $\nabla^2\hat{\Psi}(u)v$

*Given  $u$ , solve the state equation  $c(y, u) = 0$  for  $y$ .*

*Denote solution by  $y(u)$ .*

*Solve the adjoint equation  $c_y(y(u), u)^T \lambda = -\nabla_y \Psi(y(u), u)$  for  $\lambda$ .*

*Denote solution by  $\lambda(u)$ .*

*Solve  $c_y(y(u), u)w = c_u(y(u), u)v$  for  $w$ .*

*Solve  $c_y(y(u), u)^T p = \nabla_y^2 L(y(u), u, \lambda(u))w - \nabla_{yu} L(y(u), u, \lambda(u))v$  for  $p$ .*

*Compute*

$$\nabla^2\hat{\Psi}(u)v = c_u(y(u), u)^T p - \nabla_{uy} L(y(u), u, \lambda(u))w + \nabla_u^2 L(y(u), u, \lambda(u))v.$$

### 4.3.2 Hessian times Vector Product for Reservoir Problem

We now require a more elaborate notation. Setting

$$y = \begin{pmatrix} s_a^1 \\ p^1 \\ \vdots \\ p^K \end{pmatrix}, \quad u = \begin{pmatrix} q^1 \\ \vdots \\ q^K \end{pmatrix}, \quad (4.34a)$$

where  $s_a^k, p^k, q^k$  are vectors whose size is defined by the number of mesh elements and the number of wells, respectively, and defining the matrices and vectors

$$D_y^k = \begin{pmatrix} I - \Delta t D_s f^k & -\Delta t D_p f^k \\ D_s g^k & D_p g^k \end{pmatrix}, \quad Ly_{k,k-1} = \begin{pmatrix} -I & 0 \\ 0 & 0 \end{pmatrix}, \quad (4.35a)$$

$$\left( D_u^k \right)^T \Big|_{y=y(u)} = \begin{pmatrix} [-\Delta t D_q f^k]^T & [D_q g^k]^T \end{pmatrix} \Big|_{y=y(u)}, \quad \lambda^k = \begin{pmatrix} \lambda_s^k \\ \lambda_p^k \end{pmatrix}, \quad (4.35b)$$

where  $f^k$  is shorthand for  $f(t^k, s_a^k, p^k, q^k)$ , the gradient can be computed as:

$$\nabla \hat{\Psi}(u) = \Delta t \left( \begin{pmatrix} \nabla_q l^1 \\ \vdots \\ \nabla_q l^K \end{pmatrix} \right) \Big|_{y=y(u)} + \left( \begin{pmatrix} (D_u^1)^T \lambda^1 \\ \vdots \\ (D_u^K)^T \lambda^K \end{pmatrix} \right) \Big|_{y=y(u), \lambda=\lambda(u)}. \quad (4.36)$$

We need the following matrices of second order derivatives, which can be formed efficiently due to the sparsity of the mesh discretization:

$$D_{yy}^k := \begin{pmatrix} \Delta t \nabla_s^2 l^k - \Delta t \sum_{i=1}^{N_r} \nabla_s^2 f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_s^2 g_i^k \lambda_{p_i}^k & -\Delta t \sum_{i=1}^{N_r} \nabla_{sp} f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_{sp} g_i^k \lambda_{p_i}^k \\ -\Delta t \sum_{i=1}^{N_r} \nabla_{ps} f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_{ps} g_i^k \lambda_{p_i}^k & -\Delta t \sum_{i=1}^{N_r} \nabla_p^2 f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_p^2 g_i^k \lambda_{p_i}^k \end{pmatrix},$$

$$\nabla_y^2 L(y(u), u, \lambda(u)) = \left( \begin{array}{cccc} D_{yy}^1 & 0 & \cdot & \cdot \\ 0 & D_{yy}^2 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & D_{yy}^K \end{array} \right) \Big|_{y=y(u), \lambda=\lambda(u)}, \quad (4.37)$$

$$D_{uu}^k := \left( \begin{array}{c} \Delta t \nabla_q^2 l^k - \Delta t \sum_{i=1}^{N_r} \nabla_q^2 f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_q^2 g_i^k \lambda_{p_i}^k \end{array} \right),$$

$$\nabla_u^2 L(y(u), u, \lambda(u)) = \left( \begin{array}{cccc} D_{uu}^1 & 0 & \cdot & \cdot \\ 0 & D_{uu}^2 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & D_{uu}^K \end{array} \right) \Big|_{y=y(u), \lambda=\lambda(u)} \in R^{n_u \times n_u}, \quad (4.38)$$

$$D_{yu}^k := \left( \begin{array}{c} \Delta t \nabla_{sq} l^k - \Delta t \sum_{i=1}^{N_r} \nabla_{sq} f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_{sq} g_i^k \lambda_{p_i}^k \\ -\Delta t \sum_{i=1}^{N_r} \nabla_{pq} f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_{pq} g_i^k \lambda_{p_i}^k \end{array} \right),$$

$$\nabla_{yu} L(y(u), u, \lambda(u)) = \left( \begin{array}{cccc} D_{yu}^1 & 0 & \cdot & \cdot \\ 0 & D_{yu}^2 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & D_{yu}^K \end{array} \right) \Big|_{y=y(u), \lambda=\lambda(u)} \in R^{n_y \times n_u}, \quad (4.39)$$

$$D_{uy}^k := \left( \begin{array}{c} \Delta t \nabla_{qs} l^k - \Delta t \sum_{i=1}^{N_r} \nabla_{qs} f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_{qs} g_i^k \lambda_{p_i}^k \\ -\Delta t \sum_{i=1}^{N_r} \nabla_{qp} f_i^k \lambda_{s_i}^k + \sum_{i=1}^{N_r} \nabla_{qp} g_i^k \lambda_{p_i}^k \end{array} \right),$$

$$\nabla_{uy} L(y(u), u, \lambda(u)) = \left( \begin{array}{cccc} D_{uy}^1 & 0 & \cdot & \cdot \\ 0 & D_{uy}^2 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & D_{uy}^K \end{array} \right) \Big|_{y=y(u), \lambda=\lambda(u)} \in R^{n_u \times n_y}. \quad (4.40)$$

Given the matrices defined above, the Hessian-times-vector computation is defined by the following algorithm:

**Algorithm 4.3.2 (Hessian times Vector Computation IMPSAT)**

- Given  $u$ , solve the state equation  $c(y, u) = 0$  for  $y$ . Denote solution by  $y(u)$ .
- Solve the adjoint equation  $c_y(y(u), u)^T \lambda = -\nabla_y \Psi(y(u), u)$  for  $\lambda$ . Denote solution by  $\lambda(u)$ .
- For  $k = 1, \dots, K$ , solve the linearized state equation

$$D_y^k \begin{pmatrix} w_s \\ w_p \end{pmatrix}^k = D_u^k v^k + \begin{pmatrix} w_s^{k-1} \\ 0 \end{pmatrix} \text{ for } \begin{pmatrix} w_s \\ w_p \end{pmatrix}^k, \quad \text{where } w_s^0 = 0, w_p^0 = 0.$$

- For  $k = K, \dots, 1$ , solve the secondary adjoint equation

$$(D_y^k)^T \begin{pmatrix} z_s \\ z_p \end{pmatrix}^k = D_{yy}^k \begin{pmatrix} w_s \\ w_p \end{pmatrix}^k - D_{yu}^k v^k + \begin{pmatrix} z_s^{k+1} \\ 0 \end{pmatrix} \text{ for } \begin{pmatrix} z_s \\ z_p \end{pmatrix}^k,$$

$$\text{where } z_s^{K+1} = 0, z_p^{K+1} = 0.$$

- For  $k = 1, \dots, K$ , compute:

$$\left( \nabla^2 \hat{\Psi}(u) v \right)^k = (D_u^k)^T z^k - D_{uy}^k w^k + D_{uu}^k v^k.$$

In the above algorithm, it is understood that all derivative terms are evaluated at

$$y = y(u), \text{ i.e. } D_{yy}^k \big|_{y=y(u)}.$$

## 4.4 Numerical Experiments for Adjoints

In this section I investigate the impact of the time discretization and different time step sizes on the accuracy of the computed adjoint and gradient. Values for the IMPSAT discretization with the smallest time step size are used as reference.

### 4.4.1 Reservoir Model

For the numerical study, I chose the top layer of the SPE10 comparison problem as the reservoir domain. The SPE10 problem is a 3D synthetic reservoir model, which is considered to be “hard” for most commercial reservoir simulators. Its considerable size and very strong heterogeneity in the permeability makes it an ideal test case for flow based up-scaling techniques, using single phase flow, and for numerical reservoir simulation on the geological cell size level. I have chosen the top layer of the model for my study, since the heterogeneity of the model makes it an ideal candidate for the type of problem I solve. The permeability distribution and the resulting flow patterns will force the optimizer to compute an injection and production pattern that varies over time, and hence, we can expect that the dual variables and the gradient components vary strongly over time. The model uses a rectangular, structured grid of  $60 \times 220 \times 80$  cells, the top layer then being comprised of 13,200 simulation cells.

Figure 4.1 shows the permeability distribution for the 3 dimensional volume. The top layer resembles a tar bed, which models a formation close to a shoreline. Figure 4.2 shows the permeability distribution in the top layer.

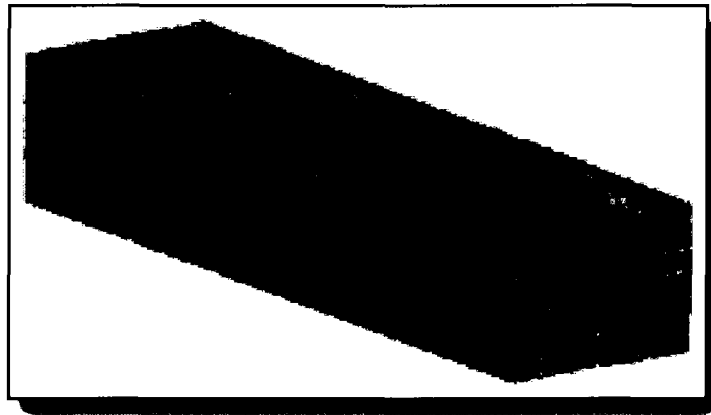


Figure 4.1: SPE10 Permeability Volume

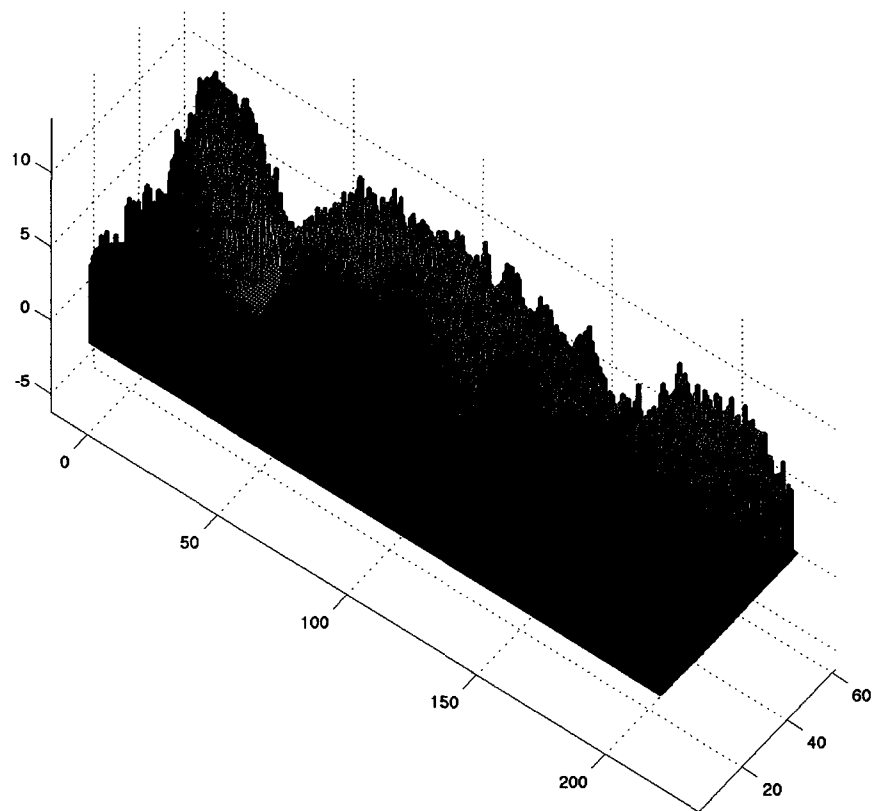


Figure 4.2: SPE10 Top Layer Permeability Distribution -  $\log \left( \frac{K_{xx} + K_{yy}}{2} \right)$



#### 4.4.2 Experimental Setup

For my numerical study, I place 4 injection and 4 production wells across the 2D reservoir model. Intentionally, these wells are not placed at optimal locations. Only two of the producers are placed in high permeability areas, and all injectors are placed in low permeability areas at the corners of the model. Figure 4.3 shows the setup. In order to stress the algorithm, I use a random time series for the injection and

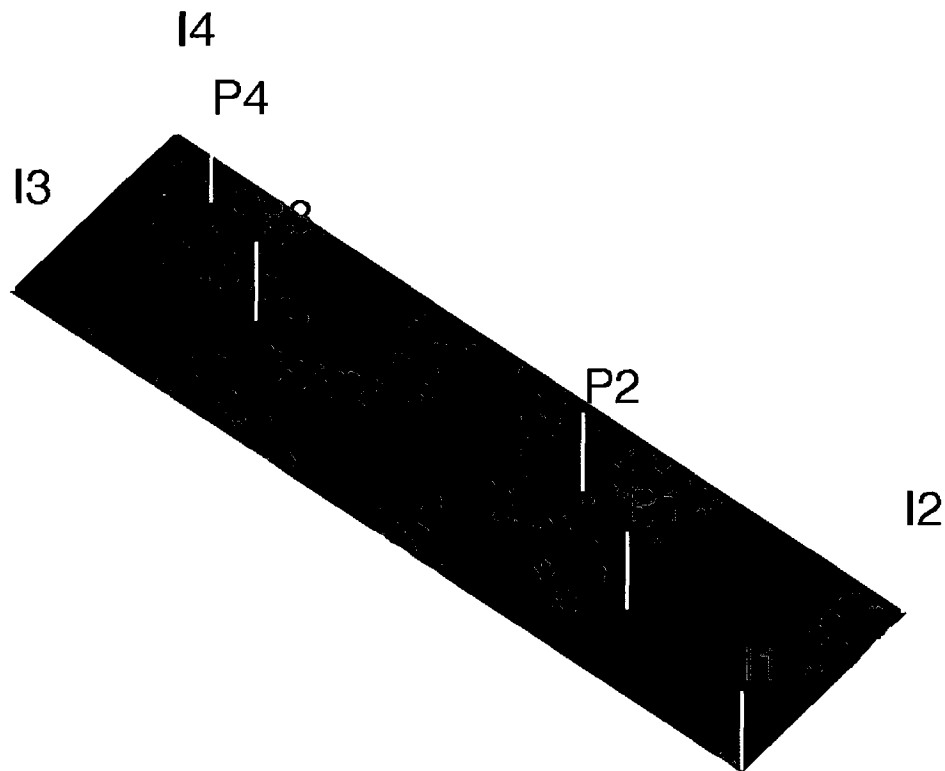


Figure 4.3: Adjoint Experimental Setup - Well Placement

production rates in the forward simulation model. The time series consist of 100 random values that are used to compute piecewise constant well rates, such that

the total injection and production is equal for all chosen time step sizes. Figure 4.4 depicts the rate profiles for the injection wells. The producers are a mirror image of these rates, since the Neumann boundary condition requires the well rates to sum up to zero.

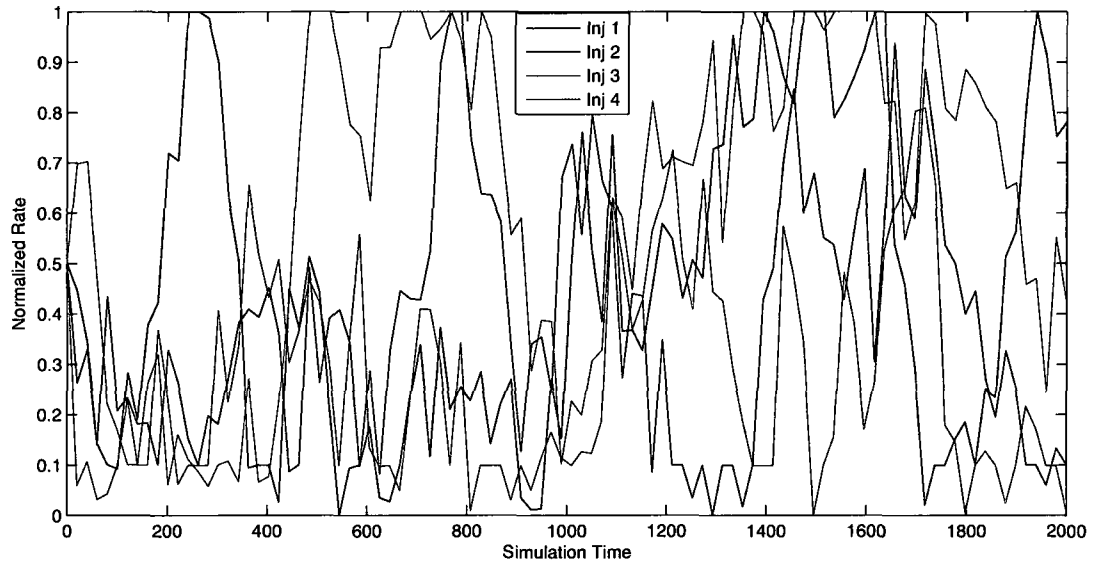


Figure 4.4: Adjoint Experimental Setup - Injection Profiles

The following collection of figures shows the gradient components for the injector and producer wells. For each well, a pair of plots is generated. The left side depicts the gradient components, as these are computed for different time step sizes using an IMPSAT forward simulation model. The right hand plot depicts the same well and time step sizes using the SEQUENTIAL formulation:

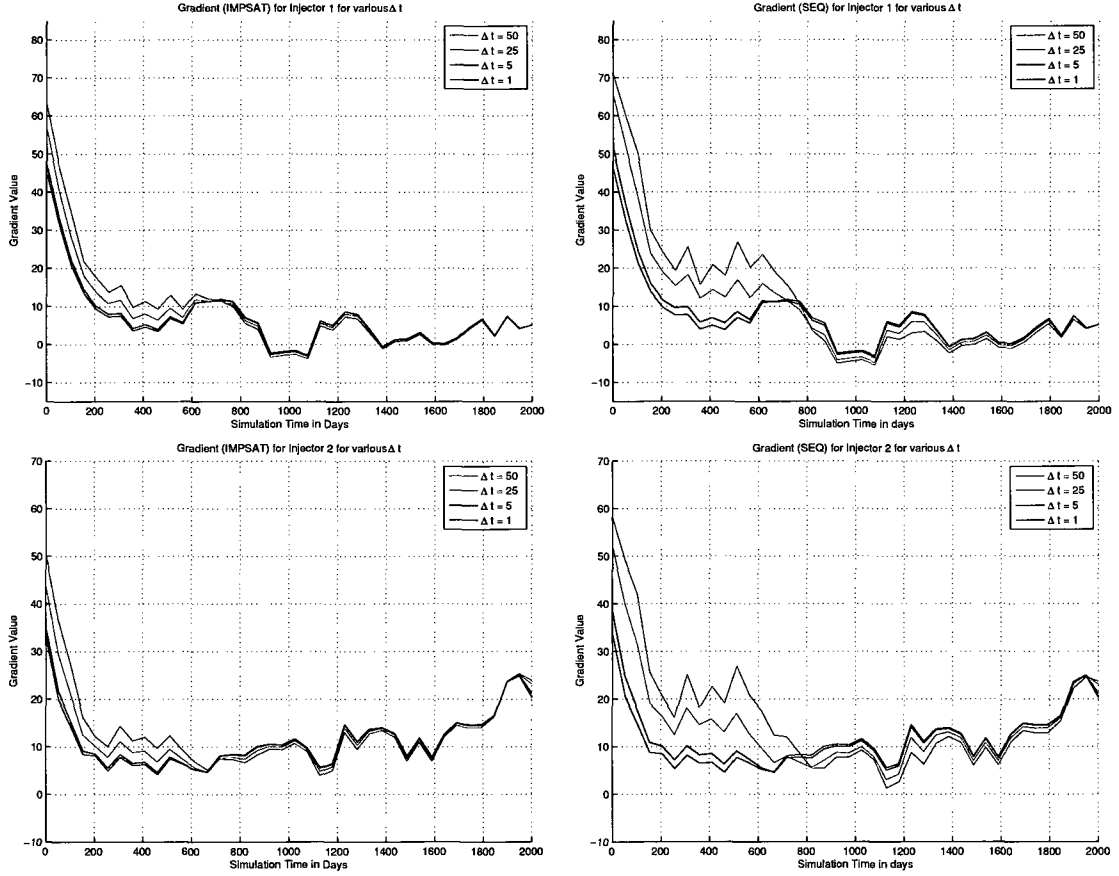


Figure 4.5: Comparison of Gradients (Injectors 1,2) IMPSAT & SEQUENTIAL

Figure 4.5 shows a trend that can be observed for all wells. Gradient components for various time step sizes match up well for later simulation times, but tend to show larger differences in the first half of the simulation run. Since the relative differences in the adjoint show the exact opposite behavior (as shown further below), the differences are most likely introduced by variations in the saturations and the partial derivatives of the discretized functions  $f(t, s(t), p(t), q(t))$  and  $g(t, s(t), p(t), q(t))$  in the forward model.

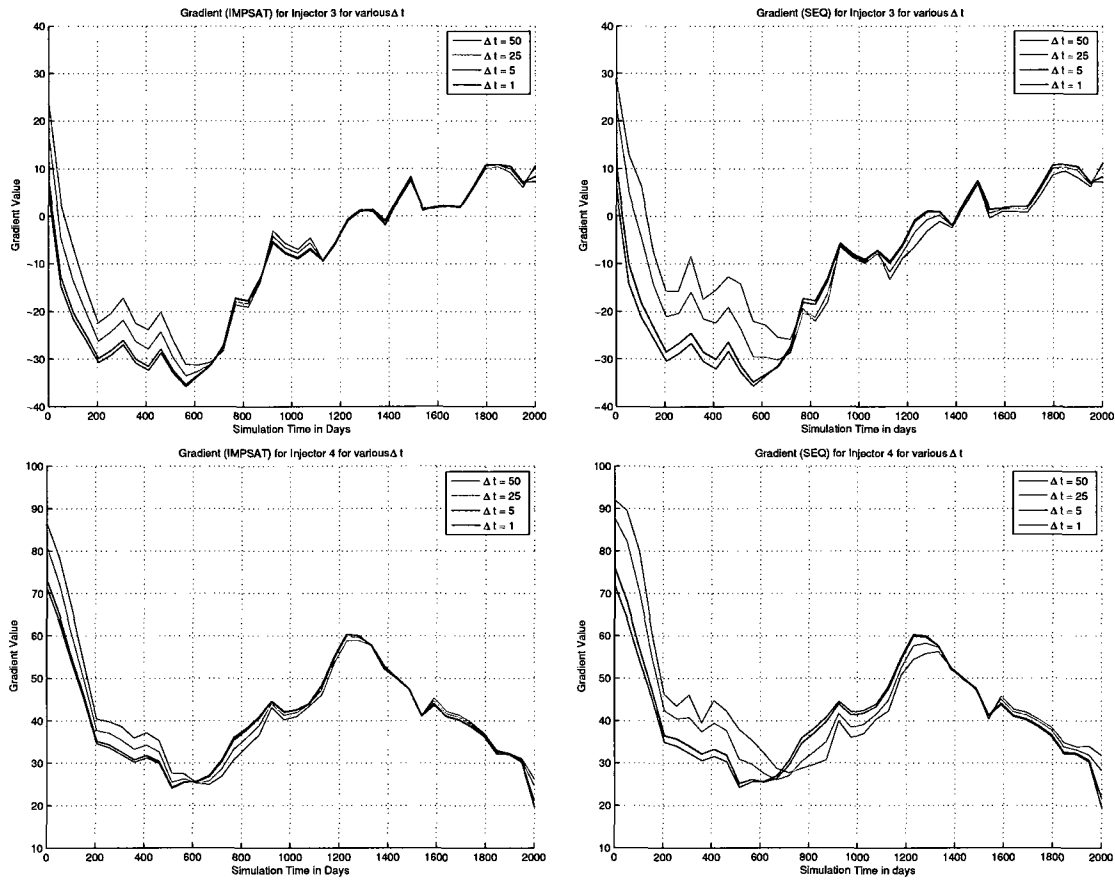


Figure 4.6: Comparison of Gradients (Injectors 3,4) IMPSAT & SEQUENTIAL

Figure 4.6 hints at a potential problem we may have to expect when using the SEQUENTIAL method for large time step sizes. The zero crossing of the gradient for injector 3 occurs at a later simulation time due to a lateral shift. While this can be attributed in part to the coarser resolution implied by larger time step sizes, the switch from a positive value to a negative value occurs at least one time step later than it would have ideally been. This may cause different results in optimization algorithms that compute step directions based on gradient information.

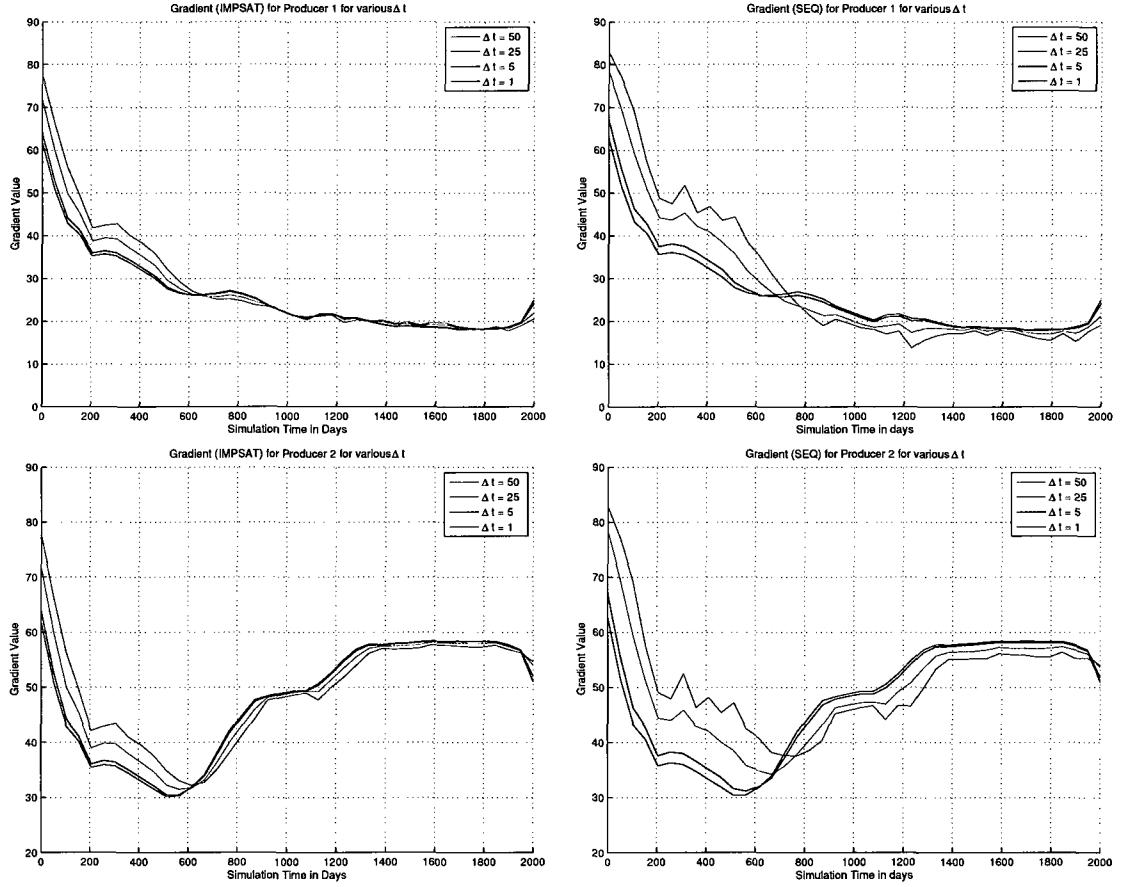


Figure 4.7: Comparison of Gradients (Producers 1,2) IMPSAT & SEQUENTIAL

Figure 4.7 shows a similar behavior for producer 1 as we have seen for injector 3. The lateral shift and difference in amplitude is actually even more pronounced. The gradient component stays above zero, however, the strong downward trend for large time steps in the SEQUENTIAL method is not observable for smaller time steps or in the IMPSAT discretization.

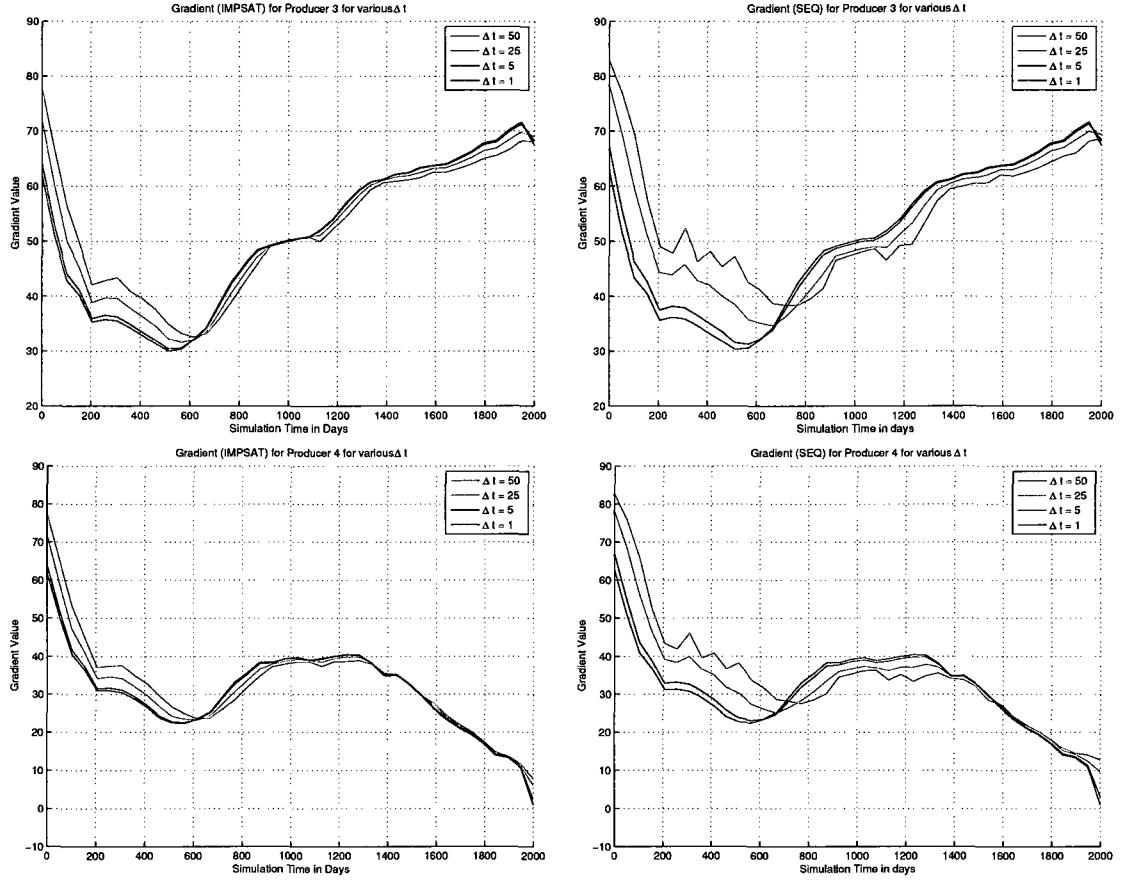


Figure 4.8: Comparison of Gradients (Producers 3,4) IMPSAT & SEQUENTIAL

The final four plots repeat the pattern observed for the injectors and the first two producers.

#### 4.4.3 Discussion of Numerical Results

As expected, the IMPSAT formulation generates gradient components for different time step sizes that are closer to each other than those generated by the SEQUENTIAL approach.

As previously mentioned, the strong lateral shift in the gradient components for the

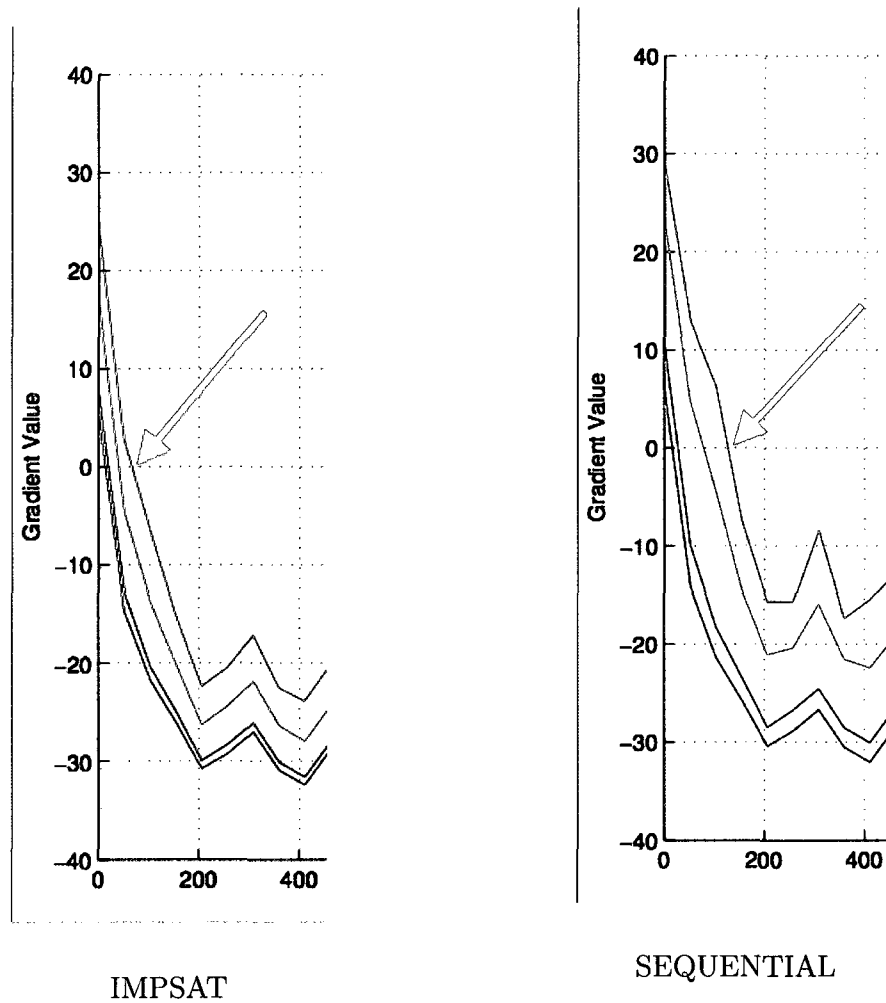
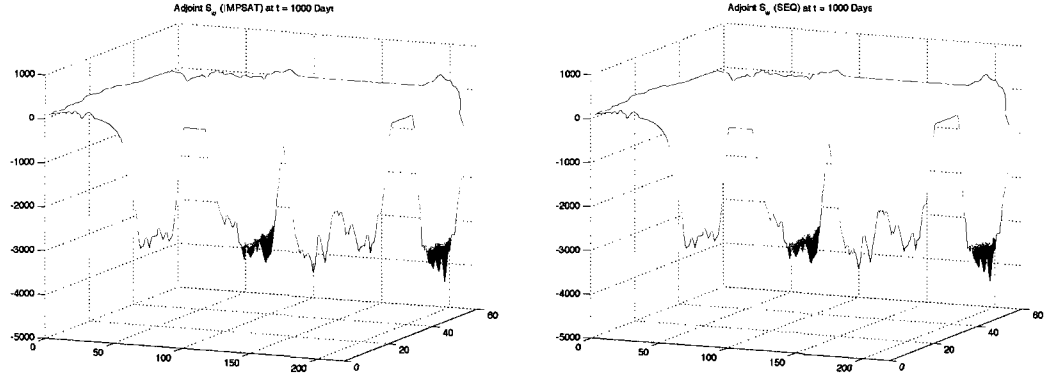


Figure 4.9: Lateral shift of Gradient for Injector 3

SEQUENTIAL formulation is a concern, and may be problematic especially when it occurs close to a zero crossing, as this is the case for injector 3. Using a small time step size, the zero crossing occurs shortly after the start of the simulation, whereas for large  $\Delta t$ , the gradient changes its sign much later. The figure above shows the problem at a higher resolution.

The following figure shows the spatial distribution of the saturation adjoint at  $t = 1000$  days. The picture is presented to give the reader an estimate for the magnitude of the components.



The following figure shows the infinity norm of the relative difference between the adjoints, computed for different time step sizes using the IMPSAT and SEQUENTIAL formulation. For each time  $t$ , the value is computed as:

$$\left\| \frac{\lambda_s^{tsp} - \lambda_s^{base}}{\|\lambda_s^{base}\|_2} \right\|_\infty,$$

where the superscript  $tsp$  indicates the adjoint computed for a specific time step size, and the superscript  $base$  indicates the baseline value, which is computed using a time step size of  $\Delta t = 1$  day.

The relative difference for the adjoint, using the sequential formulation, can reach up to 10% for large time step sizes. In addition, larger time step sizes cause instabilities in the adjoint for the second half of the simulation run. However, this does not necessarily affect the quality of the gradient. All larger discrepancies in the gradient computation for the sequential formulation occur in the first half of the simulation



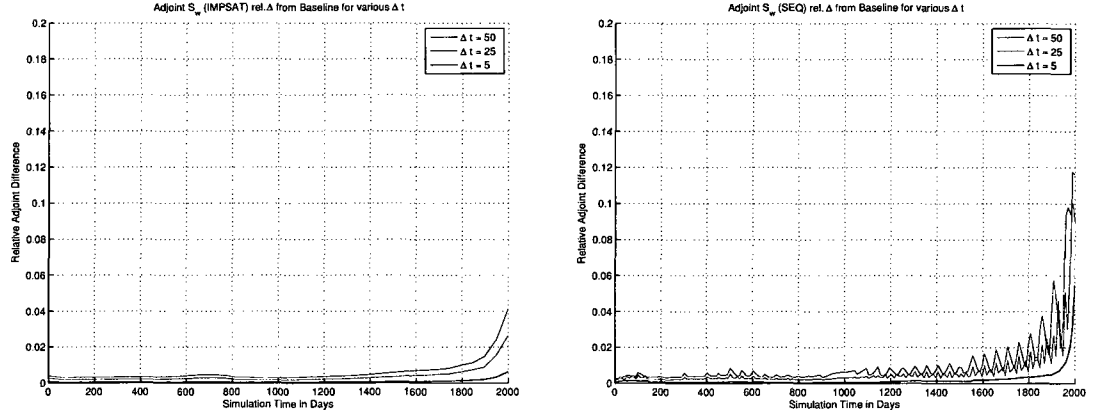


Figure 4.10: Relative Difference in Adjoint - IMPSAT and SEQUENTIAL

run, where the adjoints for different time step sizes match very well. On the other hand, gradient components for the second half match closely, whereas the adjoints show large differences.

#### 4.4.4 Conclusions

From the numerical experiments, I conclude that the sequential formulation is not as well suited for numerical reservoir optimization as the fully implicit formulation. Considering that industrial reservoir simulators, based on the volume balance formulation, often take only a single Newton iteration for the approximation of the solution of the nonlinear saturation equation, I propose to always use a fully implicit approach for the computation of the adjoint and the gradient.

## Chapter 5

# Numerical Optimization

In this chapter, I develop an active set Newton method for the solution of smooth nonlinear programs with mixed linear constraints, which I use to solve the reservoir optimization problem. The approach behind the method is based on the works of Forsgren and Murray, who describe algorithms for the solution of NLPs with linear equality constraints and linear inequality constraints [6, 7]. My adaptation of their work combines the two approaches, and considers large scale optimization problems where the Hessian is not directly available. The motivation for using this approach is to examine the usefulness of using curvature information from the Lagrangian to solve the reservoir optimization problem. Most researchers in the field have reported good results for the solution of reservoir optimization problems using methods based solely on steepest descent step computations. I want to examine if a second order method can improve upon their results. An earlier approach that I took, using an

interior point method with BFGS approximation of the Hessian, was not successful due to scaling problems of the Hessian approximation. Based on that experience, I decided to develop an optimization algorithm that makes use of analytic second order derivatives.

## 5.1 Introduction

I consider the NLP:

$$\min \quad f(x) \tag{5.1a}$$

$$\text{s.t.} \quad A_E x = b_E, \tag{5.1b}$$

$$A_I x \leq b_I, \tag{5.1c}$$

$$\text{where } x \in \mathbb{R}^n, A_E \in \mathbb{R}^{m_E \times n}, A_I \in \mathbb{R}^{m_I \times n},$$

and  $f$  is twice continuously differentiable on an open set containing the feasible points of (5.1). In my application,  $n$ ,  $m_E$ , and  $m_I$  are possibly large, and  $\nabla^2 f$  is not directly available, but products of the form  $\nabla^2 f v$  can be computed to some precision. The following two theorems state the first order necessary and second order sufficient optimality conditions for problem (5.1).

**Theorem 5.1.1** (*Karush-Kuhn-Tucker*) *If  $x^*$  is a local minimizer of (5.1), then, there exist  $\lambda^* \in \mathbb{R}^{m_E}$ ,  $\mu^* \in \mathbb{R}^{m_I}$ , such that:*

$$\nabla f(x^*) + A_E^T \lambda^* + A_I^T \mu^* = 0, \quad (5.2a)$$

$$A_E x^* - b_E = 0, \quad (5.2b)$$

$$A_I x^* - b_I \leq 0, \quad (5.2c)$$

$$(A_I x^* - b_I)^T \mu^* = 0, \quad (5.2d)$$

$$\mu^* \geq 0. \quad (5.2e)$$

The vectors  $\lambda^*$ ,  $\mu^*$  are the Lagrange multipliers associated with the linear equality and inequality constraints.

**Theorem 5.1.2** *Let  $x^*$  satisfy the KKT conditions (5.2) with Lagrange multipliers  $\lambda^*$  and  $\mu^*$ . Let  $I$  denote the set of inequality constraints, and consider the set of vectors  $D \subseteq \mathbb{R}^n$ , satisfying the following conditions:*

$$A_E d = 0, \quad a_i^T d = 0, \quad \forall i \in I : \mu_i^* > 0, \quad a_i^T d \leq 0, \quad \forall i \in I : \mu_i^* = 0.$$

*If  $d^T \nabla^2 f(x^*) d \geq 0$  for all  $d \in D$ , then  $x^*$  satisfies the second order necessary optimality conditions.*

*If  $d^T \nabla^2 f(x^*) d > 0$  for all  $d \in D$ , then  $x^*$  satisfies the second order sufficient optimality conditions.*

A discussion of first and second order optimality conditions, and proofs for theorems 5.1.1 and 5.1.2 can be found in Nocedal and Wright [13, pp. 321-341].

**Definition 5.1.3** Let  $I := \{1, 2, \dots, m_I\}$

We say that the  $i^{\text{th}}$  constraint is

active at the point  $x$  if  $a_i^T x - b_i = 0$ ,

inactive at  $x$  if  $a_i^T x - b_i < 0$ ,

violated at  $x$  if  $a_i^T x - b_i > 0$ .

Let  $x_k$  be a feasible iterate for (5.1a) with respect to the constraints (5.1b,c), and let  $I_k(x_k) \subseteq I$  be the set of active inequality constraints at the point  $x_k$ . A set  $W_k \subseteq I_k$  is called the working set of active inequality constraints at  $x_k$ . The computation of the working set will be described later.

I will use the convenient notation  $A_{W_k}$  for the working set matrix with the rows  $a_i^T$ ,  $i \in W_k$ . The matrix  $A_{W_k}$  is a sub-matrix of  $A_I$ . The matrix of all active constraints is given by:

$$A_k := \begin{bmatrix} A_E \\ A_{W_k} \end{bmatrix}. \quad (5.3)$$

Furthermore, I define  $Z_k$  to be the matrix whose columns form an orthonormal basis for  $\mathcal{N}(A_k)$ , where  $\mathcal{N}$  denotes the null space, and assume throughout that  $A_k \in \mathbb{R}^{m \times n}$  is of full row rank with  $0 < m < n$ .

## 5.2 Optimization Algorithm

In the following, I describe the computation of the step directions for the iterates, the line-search strategy used, and how the working set is computed. Since problem (5.1) is linearly constrained,  $\nabla_x^2 \mathcal{L}(x, \lambda, \mu) = \nabla^2 f(x)$ , and I use  $\nabla^2 f(x)$  throughout.

### 5.2.1 Algorithm Overview

The algorithm computes a sequence of iterates  $\{x_k\}$  such that:

$$x_{k+1} := x_k + \alpha_k p_k, \quad (5.4a)$$

$$p_k := s_k + d_k + q_k, \quad (5.4b)$$

where  $s_k$ ,  $d_k$ ,  $q_k$  are step directions, and  $\alpha_k$  is the step length, determined by a line-search procedure. In addition, a set of Lagrange multiplier estimates  $\lambda_k$ ,  $\mu_k$  is computed at each iteration. The working set is updated at the end of the line-search procedure.

#### Step Directions

The feasible step direction  $s_k$  is the Newton direction for a quadratic subproblem that I describe shortly. The vector  $q_k$  is a feasible direction aimed to remove inequality constraints from the working set, and  $d_k$  is a feasible, non-ascending direction of negative curvature. Depending on the Lagrange multiplier estimates, the current working set, and the spectrum of the reduced Hessian  $Z_k^T H_k Z_k$ , one or more of these step directions may be zero at any iteration.

Let  $E$  denote the set of equality constraints, and  $I$  the set of inequality constraints. The optimization algorithm used to solve (5.1) is outlined below. Details for the step computations are provided in the following subsections.

**Algorithm 5.2.1 (Optimization Algorithm)**

$k \leftarrow 1, x_k \leftarrow x_0, W_k \subseteq \{i \in I : a_i^T x_k - b_i = 0\}$

*while*  $k \leq \text{maxiter}$

*Compute*  $A_k, Z_k$ , *based on*  $E$  *and*  $W_k$

*Compute*  $f_k := f(x_k), \nabla f_k$

*Compute*  $s_k, \lambda_k, \mu_k$  *as described in* section 5.2.2

*if*  $Z_k^T \nabla f_k = 0$ , *test optimality and return*  $x_k$  *if optimal*

*if*  $\min(\mu_k) \geq 0$  *or*  $W_k \not\subseteq W_{k-1}$ , *set*  $q_k := 0$ , *else compute*  $q_k$  *as described in* 5.2.3

*if*  $\lambda_{\min}(Z_k^T \nabla^2 f_k Z_k) < 0$ , *compute*  $d_k$  *as described in* 5.2.4, *else set*  $d_k := 0$

$p_k = s_k + q_k + d_k$

*Perform line search and compute*  $x_{k+1} = x_k + \alpha_k p_k$  *as described in* section 5.2.5

*Compute*  $W_k^- = \{i \in W_k : a_i^T x_{k+1} - b_i = 0\}$

*Compute*  $W_k^+ \subseteq \{i \in I, i \notin W_k : a_i^T p_k > 0 \wedge a_i^T x_{k+1} - b_i = 0\}$

*Compute*  $W_{k+1} := W_k^+ \cup W_k^-$  *as described in* section 5.2.6

$k \leftarrow k + 1$

*End*

*return*  $x_k$

Algorithm 5.2.1 computes a nonzero step direction  $q_k$  only if no new active constraints have been added to the working set by the previous iteration. The intention is to make at least one feasible step within the nullspace of the current set of active constraints, before removing constraints from the working set. In the case that no feasible step  $p_k \neq 0$  can be made,  $W_{k+1} \subseteq W_k$ , and  $q_{k+1} \neq 0$  may be computed in the next iteration.

The computation of the step directions is based on the minimization of an approximate quadratic model for (5.1) at the current iterate  $x_k$ . Algorithms of this type are called *sequential quadratic programming* (SQP) methods. A description of the approach can be found in Nocedal and Wright [13, pp. 423,424]. The SQP approach can be combined with trust-region and line-search methods, of which I have chosen to use the latter. Following is a brief description of the SQP approach, tailored to NLPs with linear constraints.

Let  $x_k$  be the current iterate for NLP (5.1). Define the Lagrangian

$$\mathcal{L}(x_k, \lambda_k, \mu_k) := f(x_k) + \sum_{i \in E} (\lambda_k)_i (a_i^T x_k - b_{E_i}) + \sum_{i \in I} (\mu_k)_i (a_i^T x_k - b_{I_i}), \quad (5.5a)$$

$$\nabla_x \mathcal{L}(x_k, \lambda_k, \mu_k) := \nabla f(x_k) + A_E^T \lambda_k + A_I^T \mu_k. \quad (5.5b)$$

The SQP step can be derived in two ways:



One considers the minimization problem:

$$\min_{p_k} \quad f(x_k) + \nabla f(x_k)^T p_k + \frac{1}{2} p_k^T \nabla_x^2 f(x_k) p_k \quad (5.6a)$$

$$\text{s.t.} \quad A_k p_k = 0. \quad (5.6b)$$

The justification for (5.6) follows immediately from the second order Taylor expansion of  $f$ .

An alternative viewpoint is to solve for the KKT point using Newton's method.

$$\text{Let } \pi_k := \begin{pmatrix} \lambda_k \\ \mu_{k_j} \end{pmatrix}, \quad j \in W_k$$

be the subset of Lagrange multipliers associated with the equality constraints and the active inequality constraints in the working set. Define

$$F(x, \pi) := \begin{pmatrix} \nabla_x \mathcal{L}(x, \pi) \\ Ax \end{pmatrix} = 0,$$

where  $\mathcal{L}(x, \pi)$  is a slightly modified Lagrangian with multipliers for inactive inequality constraints removed. Linearizing around  $x_k, \pi_k$ :

$$F'(x, \pi) \Big|_{x_k, \pi_k} = \begin{pmatrix} \nabla^2 \mathcal{L}(x_k, \pi_k) & A_k^T \\ A_k & 0 \end{pmatrix},$$

which leads to the Newton step:

$$\begin{pmatrix} \nabla_x^2 f(x_k) & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} \delta x_k \\ \delta \pi \end{pmatrix} = \begin{pmatrix} -\nabla f(x_k) - A_k^T \pi_k \\ A_k x_k \end{pmatrix}. \quad (5.7)$$

Adding  $A_k^T \pi_k$  on both sides of the first equation, and using the fact that  $A_k x_k = 0$ , we obtain:

$$\begin{pmatrix} \nabla_x^2 f(x_k) & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} \delta x_k \\ \pi_{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f(x_k) \\ 0 \end{pmatrix}. \quad (5.8)$$

Nocedal and Wright [13, pp.530-533] show that the solutions to (5.8) and (5.6) are equivalent under the assumption that the matrix  $A_k$  is of full row rank, and the Hessian  $\nabla^2 f(x_k)$  is positive definite on the nullspace of the active constraints.

### 5.2.2 Computation of $s_k$

The computation of the vector  $s_k$  follows (5.8), and is identical to the approach that Forsgren and Murray use. Solve:

$$\begin{pmatrix} \hat{H}_k & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} s_k \\ \pi_k \end{pmatrix} = \begin{pmatrix} -\nabla f(x_k) \\ 0 \end{pmatrix}, \quad (5.9)$$

for  $s_k, \pi_k$ , where  $\hat{H}_k$  denotes a suitable modification of the Hessian at  $x_k$ , such that  $Z_k^T \hat{H}_k Z_k$  is positive definite.

**Lemma 5.2.2** *Let  $x_k$  be a local minimizer of (5.1), i.e.  $x_k = x^*$ , let  $W_k = I(x_k)$ , and let  $A_k$  be of full row rank. Then, the solution to (5.9) satisfies the first order KKT conditions (5.2) with  $s_k = 0$ . Furthermore, the set of Lagrange multipliers is uniquely defined.*

**Proof:** Premultiplying the first equation of (5.9) with  $Z_k^T$ , we obtain:

$$Z_k^T \hat{H}_k s_k + Z_k^T \nabla f(x_k) = 0, \quad (5.10)$$

since  $Z_k^T A_k^T \begin{bmatrix} \lambda_k \\ \mu_k \end{bmatrix} = 0$ . The second equation of (5.9) implies that  $A_k s_k = 0$ , and hence,  $s_k$  lies in the nullspace of  $A_k$ . Then there exists  $z \in \mathbb{R}^{n-m}$  such that  $s_k = Z_k z$ .

Substituting into (5.10), and using the fact that  $Z_k^T \nabla f(x^*) = 0$ , we obtain:

$$\left( Z_k^T \hat{H}_k Z_k \right) z = 0. \quad (5.11)$$

Since  $\hat{H}_k$  was assumed to be positive definite on the nullspace of the active constraints, equation (5.11) implies that  $z = 0$ , and hence, we obtain  $s_k = 0$ . Substituting  $s_k = 0$  into the first equation of (5.9) we obtain:

$$A_k^T \begin{bmatrix} \lambda_k \\ \mu_k \end{bmatrix} + Z_k^T \nabla f(x_k) = 0. \quad (5.12)$$

Since  $A_k$  is of full row rank, equation (5.12) has a unique solution. Since we required  $W_k$  to contain *all* active constraints at  $x^*$ , the solution further contains all Lagrange multipliers  $\mu_i$ , that are not required to be zero due to the complimentary conditions. Setting  $\lambda^* = \lambda_k$ ,  $\mu_i^* = (\mu_k)_i$  for  $i \in W_k$ , and  $\mu_i^* = 0$  otherwise, we obtain the result.

□

**Lemma 5.2.3** *Let  $x_k$  be a feasible point of (5.1),  $x_k \neq x^*$ , and let  $A_k$  be of full row rank. Then, a nonzero  $s_k$  from the solution to (5.9) is a descent direction that is feasible with respect to the active constraints.*

**Proof:** Premultiplying the first equation of (5.9) with  $s_k^T$ , we obtain:

$$s_k^T \widehat{H}_k s_k + s_k^T A_k^T \begin{bmatrix} \lambda_k \\ \mu_k \end{bmatrix} + s_k^T \nabla f(x_k) = 0, \quad (5.13a)$$

$$A_k s_k = 0. \quad (5.13b)$$

Clearly, (5.13b) forces  $s_k$  into the nullspace of  $A_k$ , and hence,  $s_k^T A_k^T \begin{bmatrix} \lambda_k \\ \mu_k \end{bmatrix} = 0$ . That leaves us with

$$s_k^T \widehat{H}_k s_k + s_k^T \nabla f(x_k) = 0.$$

Since, by assumption,  $\widehat{H}_k$  is positive definite on the nullspace of the active constraints, we have  $s_k^T \widehat{H}_k s_k > 0$  for  $s_k \neq 0$ , and hence:

$$s_k^T \nabla f(x_k) < 0.$$

□

### Computation of $s_k$ for Large Scale Optimization Problems

For large scale optimization problems, a direct solution of equation (5.9) is impractical, or impossible if only Hessian-times-vector products are available. In the following, I split (5.9) into two parts that can be solved separately, and suggest an iterative solution process, based on a low dimensional Krylov subspace, to solve for  $s_k$ .

Expanding (5.9) yields:

$$\widehat{H}_k s_k + A_k^T \pi_k = -\nabla f(x_k), \quad (5.14a)$$

$$A_k s_k = 0. \quad (5.14b)$$

If we define  $s_k$  to lie in the range of  $Z_k$ , i.e.  $s_k := Z_k z$ , then  $s_k$  lies in the nullspace of the active constraints, and (5.14b) is fulfilled, since  $A_k Z_k = 0$ .

**Definition 5.2.4** Let  $A_k, Z_k$  be defined as before, with  $Z_k \in \mathbb{R}^{n \times (n-m)}$ , and  $A_k \in \mathbb{R}^{m \times n}$  having full row rank with  $0 < m < n$ . Define  $Y_k \in \mathbb{R}^{n \times m}$  to be a matrix, whose columns form the basis for the orthogonal complement of the subspace spanned by the columns of  $Z_k$  in  $\mathbb{R}^n$ , and define  $X$  to be the matrix  $X := \begin{pmatrix} Z_k & Y_k \end{pmatrix} \in \mathbb{R}^{n \times n}$ . Furthermore, define  $s_k := Z_k z$ .

Pre-multiplying (5.14a) with  $X^T$  yields:

$$Z_k^T \widehat{H}_k Z_k z = -Z_k^T \nabla f(x_k) \quad (5.15a)$$

$$Y_k^T \widehat{H}_k Z_k z + Y_k^T A_k^T \pi_k = -Y_k^T \nabla f(x_k) \quad (5.15b)$$

**Definition 5.2.5** I now define  $\widehat{H}_k$  as follows: Let  $\widehat{H}_k \stackrel{\text{def}}{=} Z_k \tilde{H}_k Z_k^T$ , where  $\tilde{H}_k$  is a positive definite approximation of the reduced Hessian  $Z_k^T \nabla^2 f(x_k) Z_k$ .

Hence,  $Y_k^T \widehat{H}_k = Y_k^T Z_k \tilde{H}_k Z_k^T = 0$  (since  $Y_k^T Z_k = 0$ ), and the first term in (5.15b) cancels. This leaves us with two equations for (5.15), which can be solved separately for  $z$  and  $\pi_k$ . The solution to (5.15a) will be computed iteratively, while

the solution to (5.15b) is computed as:

$$\pi_k = - (Y_k^T A_k^T)^{-1} Y_k^T \nabla f(x_k). \quad (5.16)$$

We need a way to compute  $\tilde{H}_k$  that is compatible with the iterative solution approach.

I suggest to use low rank Lanczos factorization of the reduced Hessian:

**Definition 5.2.6** Let  $V_k T_k V_k^T$  with  $T_k \in \mathbb{R}^{l \times l}$  and  $V_k \in \mathbb{R}^{(n-m) \times l}$ , be a low rank factorization of the reduced Hessian  $Z_k^T \nabla^2 f(x_k) Z_k$ . Let  $\hat{T}_k$  denote a positive definite modification of  $T_k$ , such that  $w^T \hat{T}_k w > 0$  for all  $w \in \mathbb{R}^l$ ,  $w \neq 0$ , and define

$$\tilde{H}_k = V_k \hat{T}_k V_k^T.$$

Given the definition above, the system (5.15a) is then approximated by:

$$V_k \hat{T}_k V_k^T z = -Z_k^T \nabla f(x_k). \quad (5.17)$$

Clearly, for  $l \ll (n - m)$ ,  $\hat{H}_k$  will itself possess a very large nullspace, and hence, is only positive semi-definite on the nullspace of  $A_k$ , which is the range of  $Z_k$ . For the proof of lemma 5.2.2 we need to show that  $A_k s_k = 0$ , and that  $s_k = 0$  for  $x_k = x^*$ . The first part follows immediately from definition 5.2.4 since  $A_k Z_k = 0$ . For the second part, we use the following definition:

**Definition 5.2.7** Define  $V_k \in \mathbb{R}^{(n-m) \times l}$  to be the matrix whose column vectors  $v_j$  form an orthonormal basis for the Krylov subspace:

$$\mathcal{K}_\ell := \text{span} \left\{ Z_k^T \nabla f_k, (Z_k^T \nabla^2 f_k Z_k)^1 Z_k^T \nabla f_k, \dots, (Z_k^T \nabla^2 f_k Z_k)^{l-1} Z_k^T \nabla f_k \right\},$$

where  $f_k := f(x_k)$ . Furthermore, by orthogonality of the matrix  $Z_k$ , we have  $Z_k^T \hat{H}_k Z_k = \tilde{H}_k = V_k \hat{T}_k V_k^T$ .

**Lemma 5.2.8** *If the solution  $z$  to (5.17) is contained in the space  $\mathcal{K}_\ell$ , then  $s_k = 0$  at  $x^*$ .*

**Proof:** At  $x^*$ , we have  $Z_k^T \nabla f(x^*) = 0$ , and hence, all Krylov vectors

$$v_j = (Z_k^T \nabla^2 f_k Z_k)^{j-1} Z_k^T \nabla f_k$$

are zero. Hence,  $z$  is zero, and since  $s_k = Z_k z$ , we obtain the result.  $\square$

In the proof for lemma 5.2.3 we made use of the assumption that  $\hat{H}_k$  is positive definite on the nullspace of  $A_k$ , and hence  $s_k^T \hat{H}_k s_k > 0$  for  $s_k \neq 0$ . However, if  $\hat{H}_k$  is defined as in definition 5.2.6, then  $\hat{H}_k$  is only positive semi-definite. But we actually don't need that wide ranging assumption about  $\hat{H}_k$ ; all we require is:

$$s_k^T \hat{H}_k s_k > 0 \quad \text{for } s_k = Z_k z \neq 0.$$

**Lemma 5.2.9** *If the solution  $z$  to (5.17) is contained in the space  $\mathcal{K}_\ell$ , and if  $\hat{H}_k$  is defined as in definition 5.2.6, then  $s_k^T \hat{H}_k s_k > 0$  for  $s_k = Z_k z \neq 0$ .*

**Proof:** By construction of  $s_k$  and  $\mathcal{K}_\ell$ , we have

$$s_k^T \hat{H}_k s_k = z^T Z_k^T \hat{H}_k Z_k z, \tag{5.18a}$$

$$= z^T V_k \hat{T}_k V_k^T z. \tag{5.18b}$$

Since  $z \in \text{span } \{v_j\}$ , and all  $v_j$  are orthonormal to each other, both  $z^T V_k \neq 0$  and  $V_k^T z \neq 0$  for  $z \neq 0$ . Furthermore,  $\widehat{T}_k$  is positive definite on the space  $\mathbb{R}^l$ , and since  $z^T V_k \in \mathbb{R}^l$  and  $V_k^T z \in \mathbb{R}^l$ , we obtain the result for  $z \neq 0$ .  $\square$

In my implementation, I compute the Krylov subspace  $\mathcal{K}_\ell$  using a Lanczos iteration with  $-Z_k^T \nabla f(x_k)$  as starting vector  $v_1$ . I want to emphasize that setting  $v_1 := -Z_k^T \nabla f(x_k)$  is *essential* for the proof of lemmas 5.2.2 and 5.2.3, however, it also improves the factorization with respect to reducing the residual  $\left\| Z_k^T \nabla f(x_k) + V_k \widehat{T}_k V_k^T z \right\|$ .

### 5.2.3 Computation of $q_k$

Let  $\lambda_k, \mu_k$  be the set of Lagrange multiplier estimates computed in (5.9). My aim is to remove those inequality constraints from the working set  $W_{k+1}$  for which  $(\mu_k)_i < 0$ . For this purpose, I construct the vector  $q_k$  such that  $A_k q_k \leq 0$ , and  $a_i^T q_k < 0$  for  $i \in \mathcal{W}_k : (\mu_k)_i < 0$ . The latter implies that  $q_k$  will be feasible with respect to all equality constraints, and all those inequality constraints for which  $(\mu_k)_i \geq 0$ . The following construction process is used:

If  $\min(\mu_k) \geq 0$  or  $W_k \not\subseteq W_{k-1}$ , then  $q_k = 0$ . Otherwise, let  $0 < \vartheta \leq 1$  be given, and define the vector  $v_k \in \mathbb{R}^{m_E+m_I}$  such that:

$$\begin{aligned} v_{k_i} &= 0 & i &\in \{1, \dots, m_E\} \\ v_{k_{m_E+i}} &= -(\mu_k)_i & \text{if } (\mu_k)_i \leq \vartheta \mu_{k_{\min}}, \\ v_{k_{m_E+i}} &= 0 & \text{if } (\mu_k)_i > \vartheta \mu_{k_{\min}}. \end{aligned}$$



The direction  $q_k$  is then computed by solving the linear system:

$$\begin{pmatrix} \hat{H}_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} q_k \\ -\eta_k \end{pmatrix} = \begin{pmatrix} 0 \\ v_k \end{pmatrix}. \quad (5.19)$$

**Lemma 5.2.10** *Let  $x_k \neq x^*$  be a feasible point of (5.1), and let  $A_k$  be of full row rank. Furthermore, let  $\min(\mu_k) < 0$ . Assume that  $\hat{H}_k$  has full rank. Then, the solution to (5.19) yields a descent direction  $q_k$  that is feasible with respect to the equality constraints, and those inequality constraints  $i \in W_k$  for which  $(\mu_k)_i \geq 0$ . Furthermore, we have  $A_k q_k \leq 0$ .*

**Proof:** Expanding (5.19), we obtain:

$$\hat{H}_k q_k + A_k^T \eta_k = 0, \quad (5.20a)$$

$$-A_k q_k = v_k. \quad (5.20b)$$

Expanding (5.20b) and using the construction properties of  $v_k$  yields:

$$a_i^T q_k = 0 \quad i \in \{1, \dots, m_E\},$$

$$a_{m_E+i}^T q_k = 0 \quad i \in W_k, (\mu_k)_i \geq 0,$$

$$a_{m_E+i}^T q_k = (\mu_k)_i \quad i \in W_k, (\mu_k)_i \leq \vartheta \mu_{k_{\min}}.$$

Hence,  $q_k$  satisfies the feasibility conditions of the lemma. Since  $(\mu_k)_i \leq \vartheta \mu_{k_{\min}} < 0$ , we further have:

$$a_{m_E+i}^T q_k < 0, \quad i \in W_k \text{ and } (\mu_k)_i \leq \vartheta \mu_{k_{\min}},$$

so that  $A_k q_k \leq 0$ . The above implies that the step direction  $q_k$  aims at removing the inequality constraints  $i \in W_k$  from the working set  $W_{k+1}$ , for which  $(\mu_k)_i \leq \vartheta \mu_{k_{min}}$ .

I now show that  $q_k$  is a descent direction.

Remembering that  $\pi_k = \begin{pmatrix} \lambda_k \\ \mu_k \end{pmatrix}$  and premultiplying (5.19) by  $\begin{pmatrix} -s_k^T & \pi_k^T \end{pmatrix}$ :

$$\begin{pmatrix} -s_k^T & \pi_k^T \end{pmatrix} \begin{pmatrix} \hat{H}_k & -A_k^T \\ -A_k & 0 \end{pmatrix} \begin{pmatrix} q_k \\ -\eta_k \end{pmatrix} = \begin{pmatrix} -s_k^T & \pi_k^T \end{pmatrix} \begin{pmatrix} 0 \\ v_k \end{pmatrix}, \quad (5.21a)$$

$$\begin{pmatrix} -s_k^T \hat{H}_k - \pi_k^T A_k & 0 \end{pmatrix} \begin{pmatrix} q_k \\ -\eta_k \end{pmatrix} = \pi_k^T v_k, \quad (5.21b)$$

$$\begin{pmatrix} \nabla^T f_k & 0 \end{pmatrix} \begin{pmatrix} q_k \\ -\eta_k \end{pmatrix} = \pi_k^T v_k, \quad (5.21c)$$

$$\nabla^T f_k q_k = \pi_k^T v_k. \quad (5.21d)$$

From the construction of  $v_k$ , we have:

$$\pi_k^T v_k = - \sum_{j \in J} \mu_{k_j}^2 \leq -\mu_{k_{min}}^2, \quad J := \{j : \mu_{k_j} \leq \vartheta \mu_{k_{min}}\} \quad (5.22)$$

and substituting into (5.21d), we obtain

$$\nabla^T f_k q_k \leq -\mu_{k_{min}}^2 < 0. \quad (5.23)$$

□

### Computation of $q_k$ for large scale Problems

For the time being, we ignore the fact that  $\hat{H}_k$  will be a low rank approximation of  $\nabla^2 f(x_k)$ . Expanding (5.19) yields:

$$\hat{H}_k q_k + A_k^T \eta_k = 0,$$

$$-A_k q_k = v_k.$$

Since  $v_k$  is constructed such that  $A_k q_k \leq 0$ ,  $q_k$  is not confined to the nullspace of  $A_k$ , and will in general lie outside the range of  $Z_k$ .

**Definition 5.2.11** *Let the direction  $q_k$  be decomposed into the two vectors  $q_k := Z_k z_q + Y_k y_q$ , where  $Y_k \in \mathbb{R}^{n \times m}$  is the orthogonal complement of  $Z_k$  in  $\mathbb{R}^n$ :  $Z_k^T Y_k = 0$ .*

Plugging in, we obtain:

$$\hat{H}_k Z_k z_q + \hat{H}_k Y_k y_q + A_k^T \eta_k = 0, \quad (5.24a)$$

$$-A_k Z_k z_q - A_k Y_k y_q = v_k. \quad (5.24b)$$

The vector  $y_q$  can immediately be computed from (5.24b):

$$y_q = -(A_k Y_k)^{-1} v_k. \quad (5.25)$$

Pre-multiplying (5.24a) with  $X^T$ :

$$Z_k^T \hat{H}_k Z_k z_q + Z_k^T A_k^T \eta_k = 0, \quad (5.26a)$$

$$Y_k^T \hat{H}_k Z_k z_q + Y_k^T A_k^T \eta_k = 0. \quad (5.26b)$$

Since  $Z_k^T \widehat{H}_k Z_k$  is positive definite, it follows that  $z_q = 0$ , and we arrive at:

$$q_k = Y_k y_q = -Y_k (A_k Y_k)^{-1} v_k. \quad (5.27)$$

The terms used in the actual computation of  $q_k$  do not depend on the rank of  $\widehat{H}_k$ , or the factorization of the reduced Hessian.

#### 5.2.4 Direction of Negative Curvature $d_k$

Let  $x_k$  be a critical point of (5.1), that is,  $Z_k^T \nabla f(x_k) = 0$ .

**Definition 5.2.12** *A feasible, non-ascending direction of negative curvature at the point  $x_k$  is a vector  $d_k \in \mathbb{R}^n$  such that*

$$A_E d_k = 0, \quad (5.28a)$$

$$A_I d_k \leq 0, \quad (5.28b)$$

$$d_k^T \nabla^2 f(x_k) d_k < 0, \quad (5.28c)$$

$$\nabla^T f(x_k) d_k \leq 0. \quad (5.28d)$$

In my optimization algorithm, I use the low rank Lanczos factorization of  $Z_k^T \nabla_x^2 f(x_k) Z_k$ , described earlier, to find negative eigenvalues of the reduced Hessian. If  $\lambda_{\min}(T_k) \geq 0$ , I set  $d_k = 0$ . Otherwise, let  $u_k$  be an eigenvector of unit-norm, associated with the smallest eigenvalue of  $T_k$ . The vector  $d_k$  is then computed as

$$d_k := Z_k V_k u_k, \quad (5.29)$$

with sign chosen such that  $\nabla f(x_k)^T d_k < 0$ . The direction  $d_k$  is feasible with respect to the active constraints, since  $d_k$  lies in the range of  $Z_k$ . Since I search for negative eigenvalues of the reduced Hessian in a subspace approximation, it is possible that I compute  $d_k = 0$  in the presence of negative eigenvalues that have not been discovered by the truncated Lanczos iteration. If this happens at a saddle point of my problem, my algorithm will falsely assume that second order sufficient optimality has been achieved.

### 5.2.5 Computation of $p_k$ and Line Search Strategy

I construct  $p_k$  as the sum of the three step directions:  $p_k := s_k + d_k + q_k$ . Once  $p_k$  has been computed, an upper bound  $\alpha_{k_{max}} \geq 0$  is determined, such that  $x_k + \alpha_{k_{max}} p_k$  does not violate any inequality constraints:

$$\alpha_{k_{max}} := \min_{a_i^T p_k > 0} \left\{ \frac{b_i - a_i^T x_k}{a_i^T p_k} \right\}, \quad (5.30)$$

where  $a_i^T$  are the rows of the matrix  $A_I$ .

The starting value for  $\alpha_k$  in the line search algorithm is then computed as  $\min \{1, \alpha_{k_{max}}\}$ .

Following the example of Forsgren and Murray, I adopt the line search procedure suggested by Moré and Sorensen [11]: Select  $\alpha_k$  such that

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \beta \alpha_k \nabla f(x_k)^T p_k + \frac{1}{2} \min \{p_k^T \nabla^2 f(x_k) p_k, 0\} \alpha_k^2, \quad (5.31a)$$

and one of

$$\left| \nabla f(x_k + \alpha_k p_k)^T p_k \right| \leq \gamma \left| \nabla f(x_k)^T p_k + \min \{ p_k^T \nabla^2 f(x_k) p_k, 0 \} \alpha_k \right|, \quad (5.32a)$$

$$\alpha_k = \alpha_{k_{max}} \quad (5.32b)$$

holds. Here,  $\beta \in (0, \frac{1}{2})$  and  $\gamma \in [\beta, 1)$ . Note that for a positive definite Hessian, the procedure reduces to the strong Wolfe conditions.

**Proposition 5.2.13** *The line search parameter  $\alpha_k$  is well defined.*

**Proof:** See Moré and Sorensen [11], Lemma 5.2. □

### 5.2.6 Computation and Update of the Working Set

Given feasible  $x_0$ , the initial working set matrix  $A_0$  is assumed to have full row rank, and contain *all* constraints active at  $x_0$ . Forsgren and Murray give the following rule for updating  $W_k \rightarrow W_{k+1}$ . Define

$$W_k^- := \{i \in W_k : a_i^T p_k = 0\}, \quad (5.33a)$$

$$P_k := \{i \notin W_k : a_i^T p_k > 0 \text{ and } a_i^T x_{k+1} = b_i\}, \quad (5.33b)$$

$$W_k^+ \subseteq P_k. \quad (5.33c)$$

The updated working set  $W_{k+1}$  is then computed as:

$$W_{k+1} := W_k^+ \cup W_k^-, \quad (5.34)$$

where  $W_k^+$  was chosen such that  $A_{k+1}$  is of full rank.

### 5.3 Strategy for Problems with Large Working Sets

The algorithm developed so far has the drawback that at most a few constraints are added to the working set at each iteration. This is a consequence from the computation of  $\alpha_{k_{max}}$  in the line-search algorithm, which scales the direction  $p_k$  such that the point  $x_k + \alpha_{k_{max}}p_k$  lies exactly on the boundary of the feasible set, unless the direction  $p_k$  is unconstrained. In the context of simulation based optimization with a large number of active constraints at the solution  $x^*$ , this behavior poses a serious problem, since each iteration is very costly. It is therefore desirable to find a starting vector  $x_0$  which is sufficiently close to the solution  $x^*$ , and produces an initial working set  $W_0$  that contains already most of the active constraints for  $x^*$ . I obtain the starting vector  $x_0$  by solving (5.1) approximately, using a *projected gradient* algorithm.

#### 5.3.1 Gradient Projection Method

Gradient projection methods were originally proposed by Goldstein, Levitin and Polyak, see D. B. Bertsekas [5] for a reference. The gradient projection algorithm projects a steepest descent step back onto the feasible set. By varying the size of the steepest descent step (using line search), the projection moves along a path that is called the *projection arc*. If the unconstrained minimum lies outside the feasible set  $\mathcal{F}$ , and  $\mathcal{F}$  is convex, the path of the projection arc will follow the boundary of  $\mathcal{F}$ . I first describe the method in general, and then give specifics for the adaptation to my

optimization problem.

Consider the NLP:

$$\min f(x) \tag{5.35a}$$

$$\text{s.t.: } x \in \mathcal{F}, \tag{5.35b}$$

where  $x \in \mathbb{R}^n$ ,  $\mathcal{F}$  is a nonempty, closed, and convex set, and  $f$  is continuously differentiable on an open set that contains  $\mathcal{F}$ . Let

$$x_k(\alpha) := \mathcal{P}(x_k - \alpha \nabla f(x_k), \mathcal{F}),$$

where  $\mathcal{P}(x, \mathcal{F})$  denotes the projection of  $x$  onto the feasible set  $\mathcal{F}$ . Using the Euclidean 2-norm,  $\mathcal{P}$  is defined by

$$\mathcal{P}(x, \mathcal{F}) \stackrel{\text{def}}{=} \operatorname{argmin} \{ \|z - x\|_2 : z \in \mathcal{F} \}.$$

If  $x_k = x_k(0)$  is a feasible iterate for problem (5.35),  $x_k \neq x^*$ , the projected gradient method defines the next iterate as

$$x_{k+1} := x_k(\alpha_k) \quad \text{for } \alpha_k > 0,$$

where  $\alpha_k$  is determined by the line search rule. Bertsekas [5] proposes a practical finite procedure to determine the step size  $\alpha_k$ :

Given  $\beta, \mu$  in  $(0, 1)$ , and  $\gamma > 0$ , he suggests to use an Armijo rule where

$$\alpha_k := \beta^{m_k} \gamma,$$



and where  $m_k$  is the smallest nonnegative integer such that

$$f(x_{k+1}) \leq f(x_k) + \mu (\nabla^T f(x_k) (x_{k+1} - x_k)).$$

Under the given assumptions about  $f$  and  $\mathcal{F}$ , Bertsekas shows that limit points of the sequence  $\{x_k\}$  are stationary points of (5.35), and that if  $\{x_k\}$  converges to a local minimizer that satisfies the second order sufficient conditions and strict complementary, then the set of active constraints is identified in a finite number of steps.

### Computation of the projector for my optimization problem

If  $x_k - \alpha_k \nabla f(x_k)$  is feasible, then  $\mathcal{P} = I$ . Otherwise, let  $E = \{1, 2, \dots, m_E\}$  denote the set of linear equality constraints, and let  $I = \{1, 2, \dots, m_I\}$  denote the set of linear inequality constraints as before. I compute  $\mathcal{P}(x, \mathcal{F})$  by solving the minimization problem:

$$\min_x \|x - (x_k - \alpha_k \nabla f(x_k))\|_2 \quad (5.36a)$$

$$\text{s.t. } A_E x = b_E, \quad (5.36b)$$

$$A_I x \leq b_I. \quad (5.36c)$$

Finding a solution to problem (5.36) may pose difficulties for large  $n$ ,  $m_E$ , and  $m_I$ .

An alternative is to search for  $\delta_x$  in the nullspace of the equality constraints. Setting:

$$x = x_k + \delta x, \quad (5.37a)$$

$$\delta x = Z_E v, \quad (5.37b)$$

where  $Z_E$  is a matrix with orthonormal columns forming a basis for the nullspace of  $A_E$ , I can reformulate the problem into:

$$\min_v \quad \frac{1}{2}v^T v + \frac{1}{2}v^T Z_E^T (x_k + \alpha_k \nabla f(x_k)) \quad (5.38a)$$

$$\text{s.t.} \quad A_I Z_E v \leq b_I - A_I x_k. \quad (5.38b)$$

### Stopping Criteria for Gradient Projection Method

I use a simple stopping criteria for the gradient projection method, since my goal is to find an initial guess for the active set Newton method, which is my main optimization algorithm. The iteration is stopped whenever one of the following three conditions is true:

The norm of the reduced gradient  $\|Z_k^T \nabla f(x_k)\|_2$  is smaller than a given tolerance;

The number of iterations exceeds the allowed maximum;

The minimum step size is reached in the line search.

In order to compute the reduced gradient, the working set of active constraints has to be computed for  $x_k$ . For my implementation of the gradient projection method, the working set  $W_k$  is always identical to the set of active constraints:

$$W_k = \{i : a_i^T x_k - b_i = 0\}, \quad (5.39a)$$

$$A_k = \begin{pmatrix} A_E \\ A_I^k \end{pmatrix}. \quad (5.39b)$$

The following outline describes the gradient projection algorithm:

**Algorithm 5.3.1 (Gradient Projection Algorithm)**

$k \leftarrow 1, x_k \leftarrow x_0, W_k = \{i : a_i^T x_k - b_i = 0\}$

*Compute  $Z_E$*

*while  $k \leq \text{maxiter}$*

*Compute  $A_k, Z_k$  based on  $E$  and  $W_k$*

*Compute  $f_k := f(x_k), \nabla f_k$*

*if  $Z_k^T \nabla f_k \approx 0$ , return  $\bar{x} := x_k$*

*$m_k \leftarrow 1, \alpha_k \leftarrow \beta^{m_k} \gamma$*

*while  $\alpha_k \geq \alpha_{\min}$*

*Compute  $x_k(\alpha_k)$  using the projector described in section 5.3.1*

*if  $f(x_k(\alpha_k)) \leq f(x_k) + \mu (\nabla^T f(x_k)(x_k(\alpha_k) - x_k)) \Rightarrow \text{break}$*

*$m_k \leftarrow m_k + 1, \alpha_k \leftarrow \beta^{m_k} \gamma$*

*End*

*if  $\alpha_k < \alpha_{\min}$ , return  $\bar{x} := x_k$*

*$x_{k+1} \leftarrow x_k(\alpha_k)$*

*Compute  $W_{k+1} := \{i \in I : a_i^T x_{k+1} - b_i = 0\}$*

*$k \leftarrow k + 1$*

*End*

*return  $\bar{x} := x_k$*

### 5.3.2 Integrated Optimization Approach

As mentioned at the begin of this section, I first obtain an approximate solution  $x_0$  to (5.1) using the gradient projection algorithm, and feed this value as the starting point to the active set Newton algorithm. A variation of the technique, which I have used successfully, is to obtain the approximate solution  $x_0$  on a coarser time-scale, interpolate the result onto a finer time-grid, and continue the solution process with the active set Newton method.

## 5.4 Numerical Optimization Results

In this section I present the numerical results for solving the reservoir optimization problem for three different experiments. The first two experiments consider a simple box model as reservoir, while the third experiment uses the top layer of the SPE10 comparison case as the reservoir domain.

### 5.4.1 Experiment 1

I use a simple box model that consists of 625 finite volumes, arranged in a 25x25 quadratic grid. For this experiment, I use an isotropic permeability tensor with a single high permeability column on one side of the grid. A total of 3 injector wells and one producer is laid out such that one of the injectors is connected with the producer via the high permeability column. Figure 5.1 shows the setup: My expectation is that the optimization algorithm will reduce or shut-in injector 2 in order to avoid an early water break-through that is penalized by the cost function.

#### Simulator and Optimizer Settings

For this experiment, I chose a simulation period of 1,000 days with a time step size of  $\Delta t = 50$  days. I compute an initial guess for the active set Newton method by taking 1 iteration of the projected gradient algorithm. For the active set method, I chose a KKT norm of 0.01 as target, and require the algorithm to reach second order sufficient optimality conditions.



Figure 5.1: Optimization Experimental Setup 1

### Iteration Results

Following is the iteration information for both optimization algorithms. Due to the large number of iterations taken in the active set method, I show individual rows for the first 5 iterations, followed by increments of 25 iterations for the remainder of the optimization process.

The first table lists iteration information for the gradient projection method. The objective function value  $f(x_k)$  and the reduced gradient  $Z^T \nabla f(x_k)$  are evaluated at the begin of the iteration, using the rate and constraint information from the previous iteration or the initial rate setting. Step norms and the number of active constraints

Table 5.1: Iteration Results for Gradient Projection Algorithm - Experiment 1

Iteration	$f(x_k)$	$\ x_{k+1} - x_k\ _2$	$\ Z^T \nabla f(x_k)\ _2$	$ I(x_k) $
1	-3.96e5	1	2.42e2	20
2	-5.09e5	7.45e-09	1.15e+1	40

are end-of-iteration values, as determined by the Armijo rule. For this experiment, the initial rate setting was a uniform pattern of unit well rates for injectors and producers alike. The algorithm terminated after 1 iteration (which was the preset maximum number of allowed iterations). The rows of the second table depict the iteration information for the active set Newton method, and are to be interpreted as follows: The values for the objective function  $f(x_k)$ , the KKT norm  $\|KKT\|_2$ , and the working set size  $Size\ W_k$  are beginning-of-iteration values. The column  $\max\{-\mu, 0\}$  depicts the largest negative Lagrange multiplier encountered (for active inequality constraints present in the working set  $W_k$ ), and the column  $\max\{-\sigma, 0\}$  depicts the largest negative eigenvalue computed for the reduced Hessian, using the Lanczos factorization.

Table 5.2: Iteration Results for Active Set Newton Algorithm - Experiment 1

Iteration	$f(x_k)$	$\ x_{k+1} - x_k\ _2$	$\ KKT\ _2$	$\max\{-\mu, 0\}$	$\max\{-\sigma, 0\}$	$ W_k $
1	-5.093e5	6.79e-2	1.15e+1	-0	0	40
2	-5.093e5	2.12e+0	1.12e+1	9.2e+0	0	40
3	-5.096e5	6.05e-1	1.06e+1	4	0	39
4	-5.096e5	6.50e-2	1.07e+1	0	0	38
5	-5.097e5	6.37e-2	1.04e+1	0	0	38
25	-5.101e5	4.21e-2	6.97e+0	0	0	38
50	-5.103e5	2.52e-2	4.21e+0	0	0	38
75	-5.104e5	1.51e-2	2.54e+0	0	0	38
100	-5.104e5	9.10e-3	1.53e+0	0	0	38
125	-5.105e5	5.50e-3	9.25e-1	0	0	38
150	-5.105e5	3.30e-3	5.58e-1	0	0	38
175	-5.105e5	2.00e-3	3.37e-1	0	0	38
200	-5.105e5	1.20e-3	2.03e-1	0	0	38
225	-5.105e5	7.29e-4	1.23e-1	0	0	38
250	-5.105e5	4.40e-4	7.41e-2	0	0	38
275	-5.105e5	2.65e-4	4.47e-2	0	0	38
300	-5.105e5	1.60e-4	2.70e-2	0	0	38
325	-5.105e5	9.67e-05	1.63e-2	0	0	38
350	-5.105e5	0	9.83e-3	0	0	38



Following are figures that graph selected iteration information for the active set Newton algorithm. The algorithm was able to continuously decrease the KKT norm towards the given target. However, it can be seen from the plot of the objective function values in figure 5.4 that the target KKT norm was too small, and many unnecessary iterations have been taken.

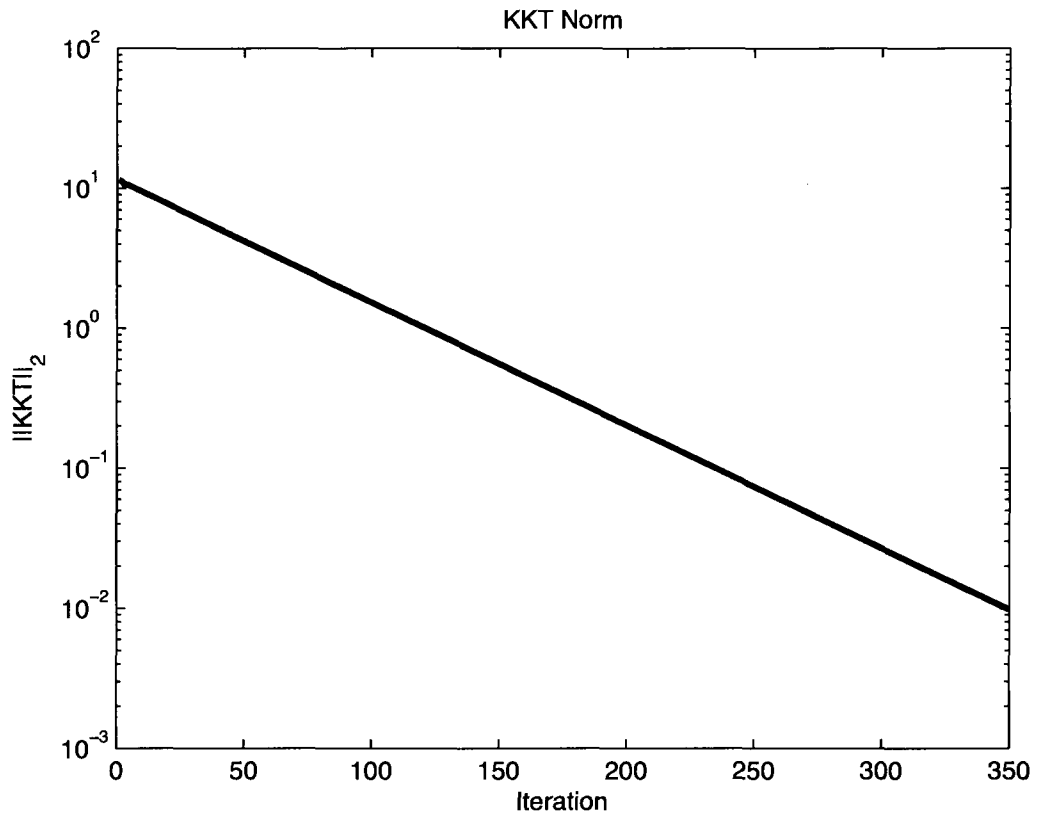


Figure 5.2: KKT Norm/Iteration - Experiment 1

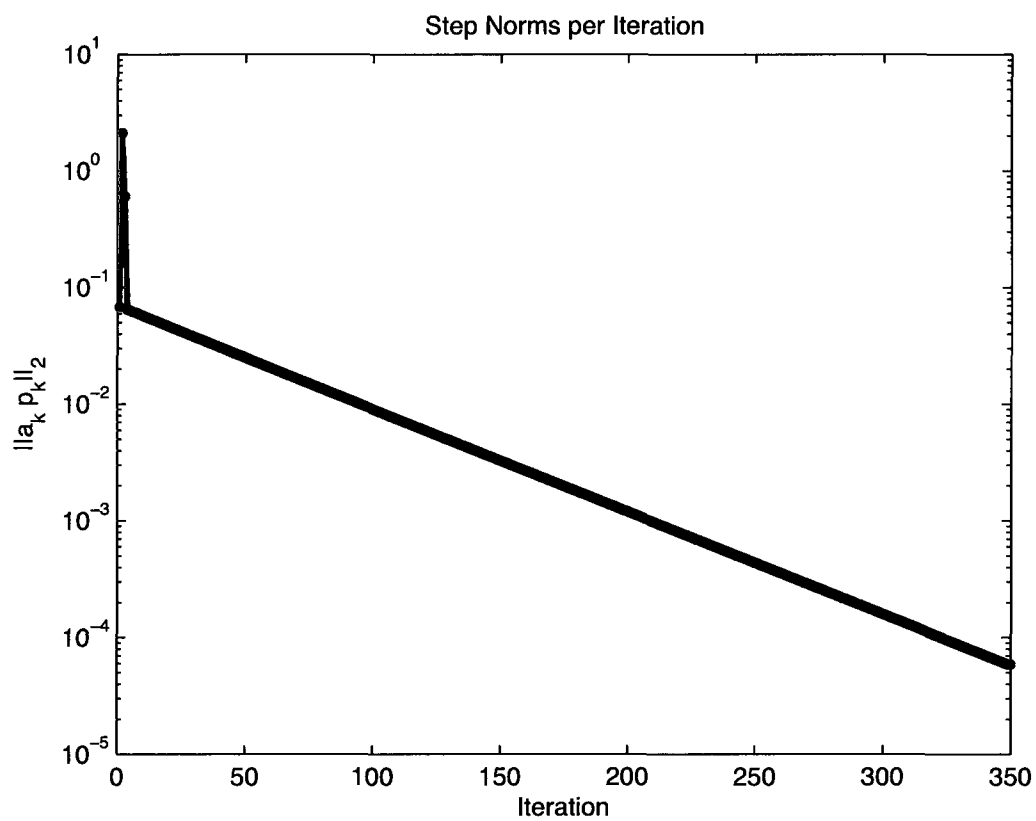


Figure 5.3: Step Norm/Iteration - Experiment 1

The following figure depicts the decrease of the objective function value during the iterations taken in the active set Newton method.

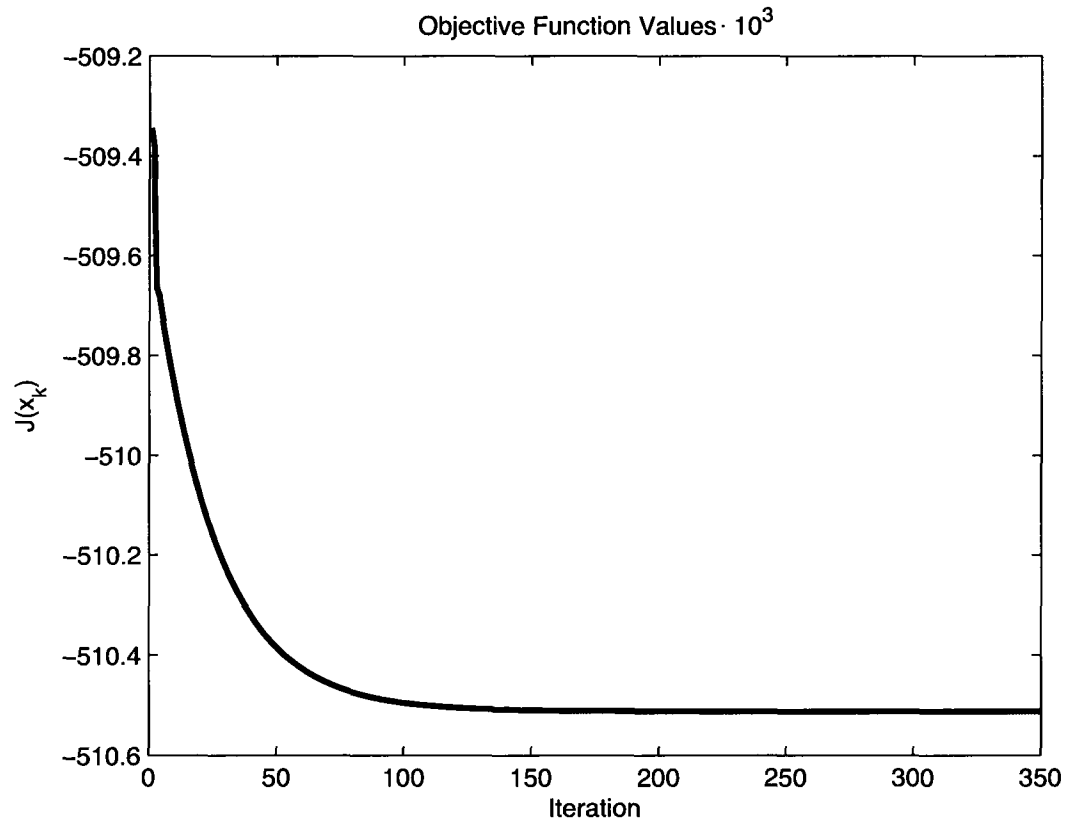


Figure 5.4: Objective Function Values - Experiment 1

The next figure plots the computed optimal well rates. As expected, the second injector is shut-in for most of the simulation period. However, close to the end of the simulation it is activated and able to push some oil from the high permeability column towards the producer.

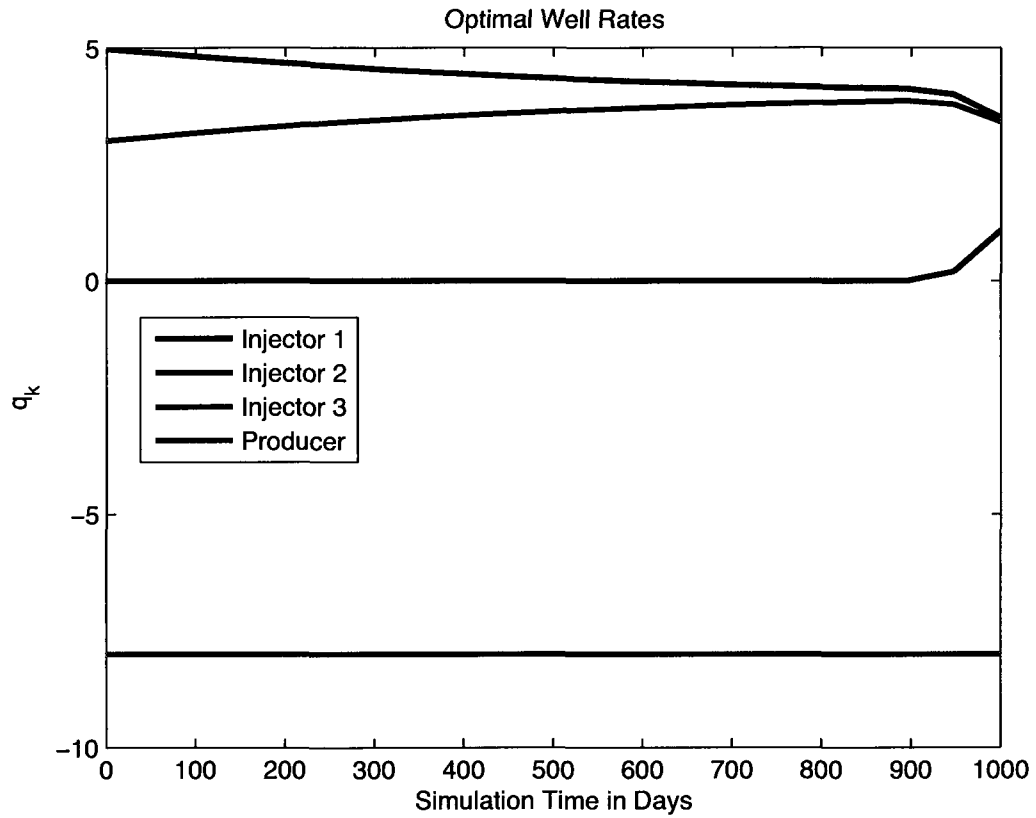


Figure 5.5: Computed Optimal Well Rates - Experiment 1

The last figure is a plot of the final aqueous saturation profile after 1,000 days of injection. The profile clearly depicts how the optimizer steered the fluids towards the producer, avoiding the high permeability column.

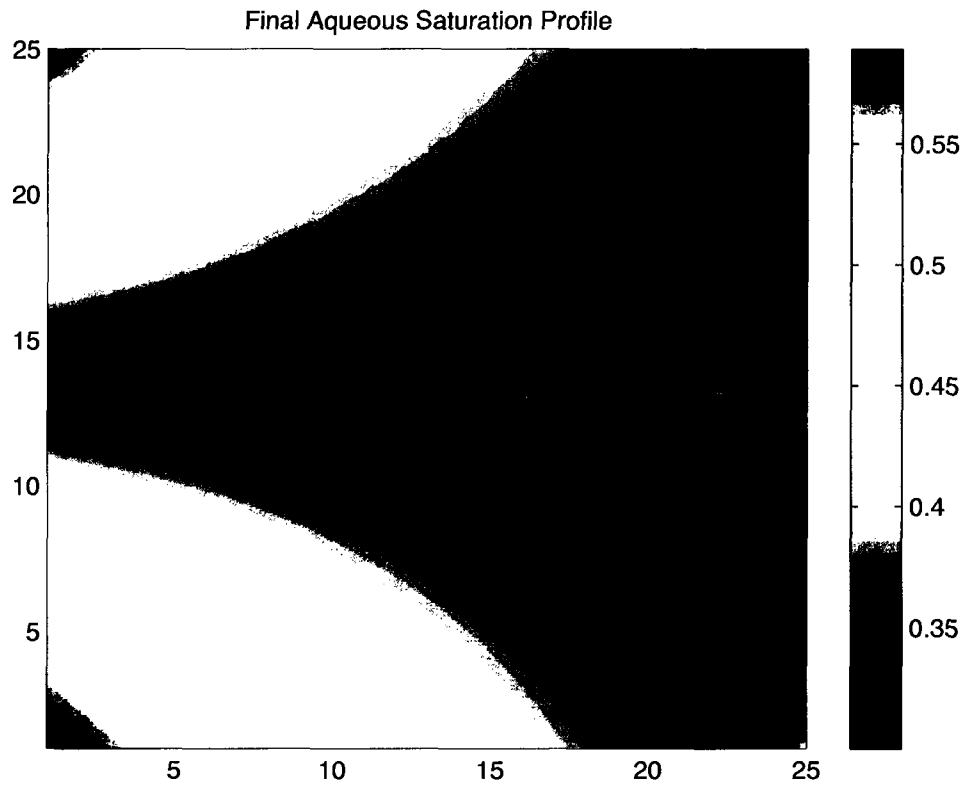


Figure 5.6: Final Aqueous Saturation Profile - Experiment 1

### 5.4.2 Experiment 2

For the second experiment, I use an unisotropic permeability tensor with

$$K_1(x) = 100K_2(x)$$

on the same grid that I used in the first experiment. The layout of the three injectors and the two producer wells is such that injector 2 will most likely force an early water breakthrough in producer 2, injector 1 will least likely force an early water breakthrough in any producer, and injector 3 may force an early water breakthrough in any producer, and injector 3 may force an early water breakthrough in producer 1. Figure 5.7 shows the layout of the wells:

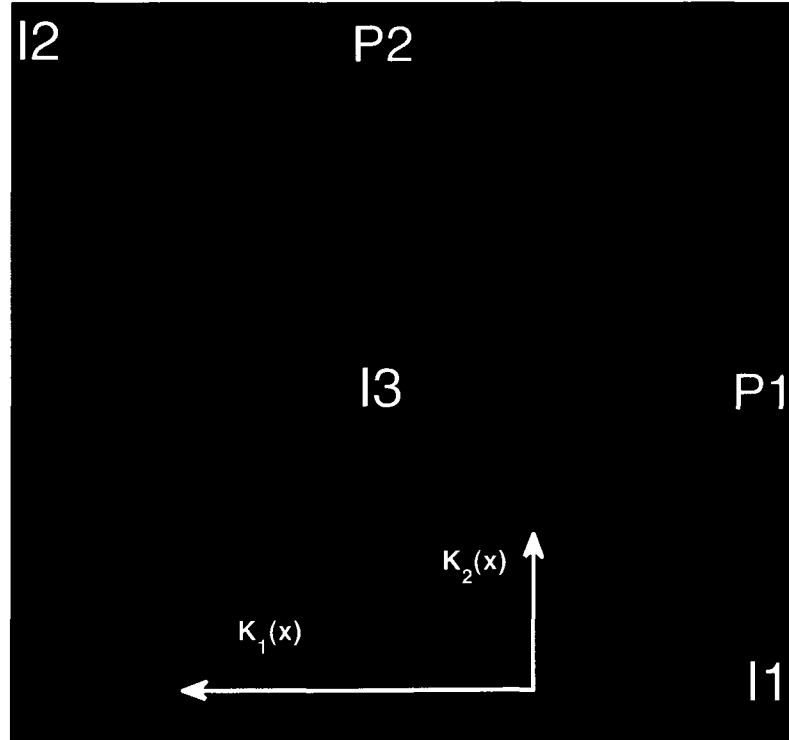


Figure 5.7: Optimization Experimental Setup 2

My expectation is that the optimization algorithm will allocate well rates such that injector 1 will inject the most water, followed by injector 3 and injector 2.

### Simulator and Optimizer Settings

For this experiment, I chose a simulation period of 1,500 days with a time step size of  $\Delta t = 50$  days. I compute an initial guess for the active set Newton method by taking 3 iterations of the projected gradient algorithm. For the active set method, I chose a KKT norm of  $5 \cdot 10^{-1}$  as target, based on the experience from the first experiment.

### Iteration Results

Following is the iteration information for both optimization algorithms. Due to the large number of iterations taken in the active set method, I show individual rows for the first 10 iterations, followed by increments of 25 iterations for the remainder of the optimization process.

Table 5.3: Iteration Results for Gradient Projection Algorithm - Experiment 2

Iteration	$f(x_k)$	$\ x_{k+1} - x_k\ _2$	$\ Z^T \nabla f(x_k)\ _2$	$ I(x_k) $
1	-2.67e5	1	3.42e2	0
2	-7.02e5	3.73e-09	4.21e+1	90
3	-7.02e5	9.31e-10	4.21e+1	90
4	-7.02e5	1.16e-10	4.21e+1	90

Table 5.4: Iteration Results for Active Set Newton Algorithm - Experiment 2

Iteration	$f(x_k)$	$\ x_{k+1} - x_k\ _2$	$\ KKT\ _2$	$\max\{-\mu, 0\}$	$\max\{-\sigma, 0\}$	$ W_k $
1	-7.022e5	1.14e-1	4.21e+1	0	0	90
2	-7.024e5	3.07e-1	3.60e+1	0	0	91
3	-7.029e5	2.56e-07	3.52e+1	7.40e+1	0	91
4	-7.029e5	3.27e+0	3.52e+1	3.00e+1	0	91
5	-7.036e5	2.83e+0	3.70e+1	1.20e+1	0	90
25	-7.124e5	2.10e-1	3.39e+1	0	0	72
50	-7.174e5	1.48e+0	2.10e+1	3.10e-2	4.20e-1	71
75	-7.190e5	1.08e-1	1.55e+1	0	0	73
100	-7.201e5	7.00e-2	9.34e+0	0	0	73
125	-7.205e5	4.70e-2	5.64e+0	0	0	73
150	-7.207e5	0	3.72e+0	6.90e+1	0	75
175	-7.207e5	2.36e-2	2.44e+0	0	0	75
200	-7.208e5	1.46e-2	1.47e+0	0	0	75
225	-7.208e5	1.32e-2	8.96e-1	0	0	75
250	-7.208e5	8.37e-3	5.41e-1	1.90e-3	0	75
256	-7.208e5	0	4.91e-1	0	0	74



Following are figures that graph selected iteration information for the active set Newton algorithm. This time, the algorithm required 25 iterations to sweep off unwanted constraints from the working set, before it was able to continuously decrease the KKT norm towards the given target.

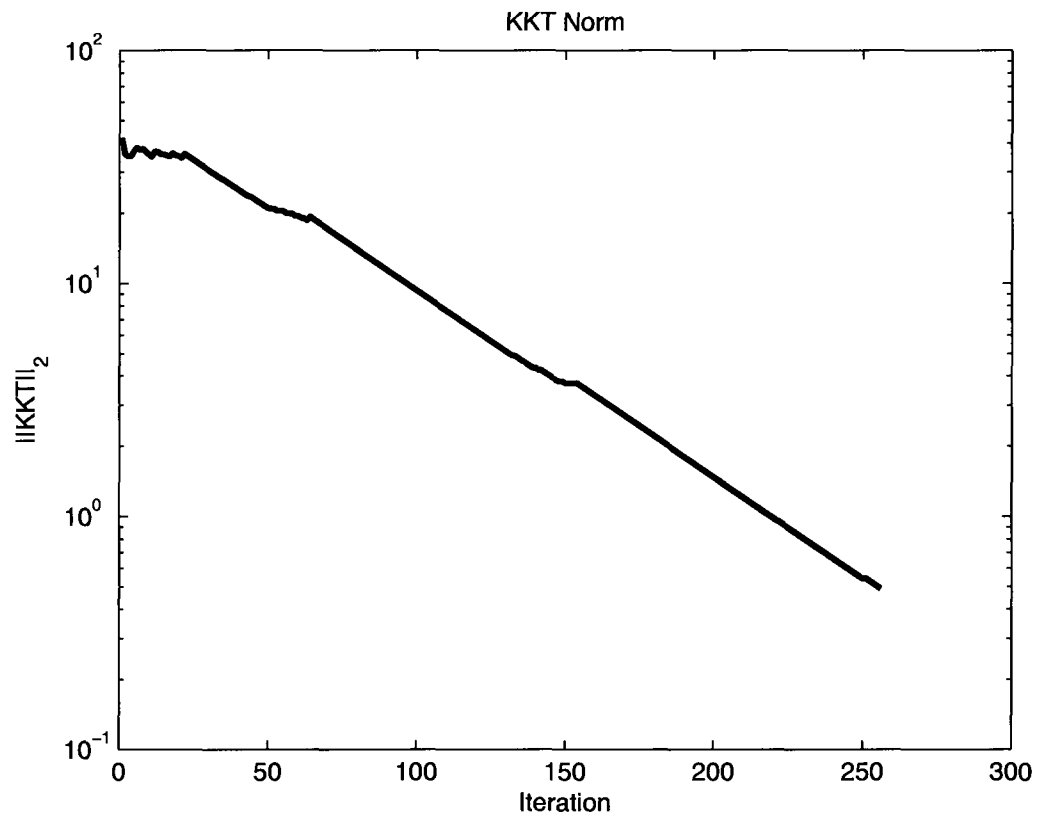


Figure 5.8: KKT Norm/Iteration - Experiment 2

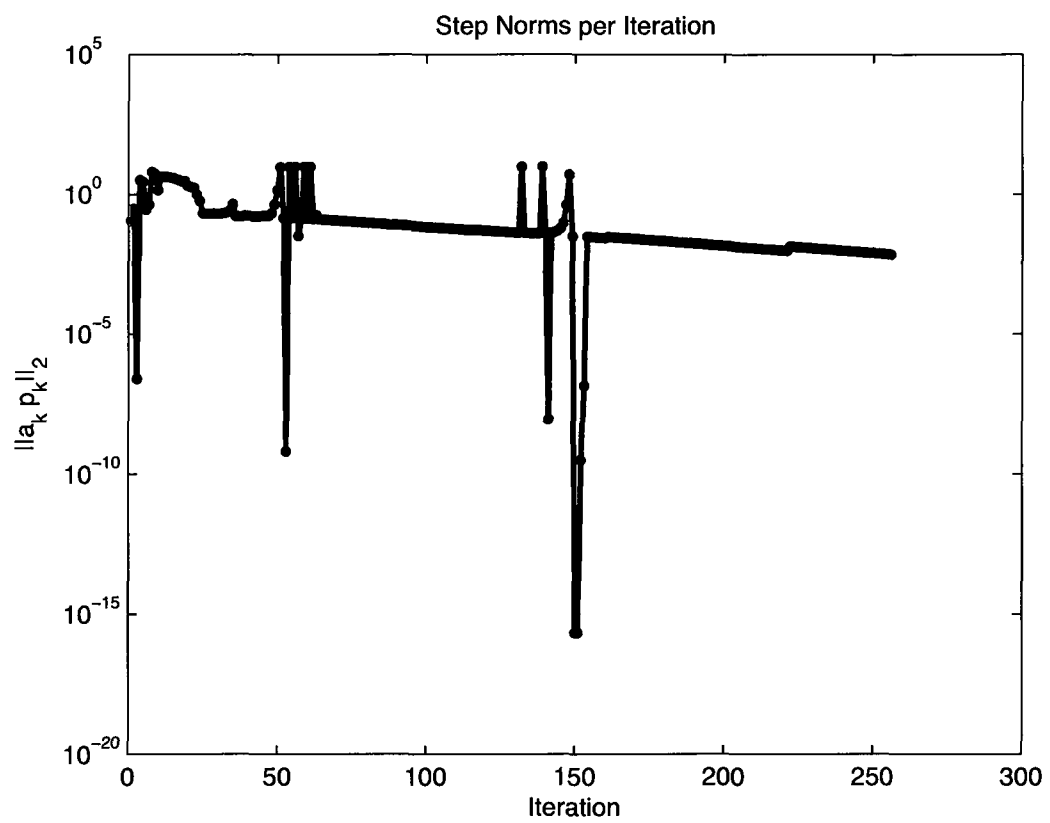


Figure 5.9: Step Norm/Iteration - Experiment 2

The following figure depicts the decrease of the objective function value during the iterations taken in the active set Newton method.

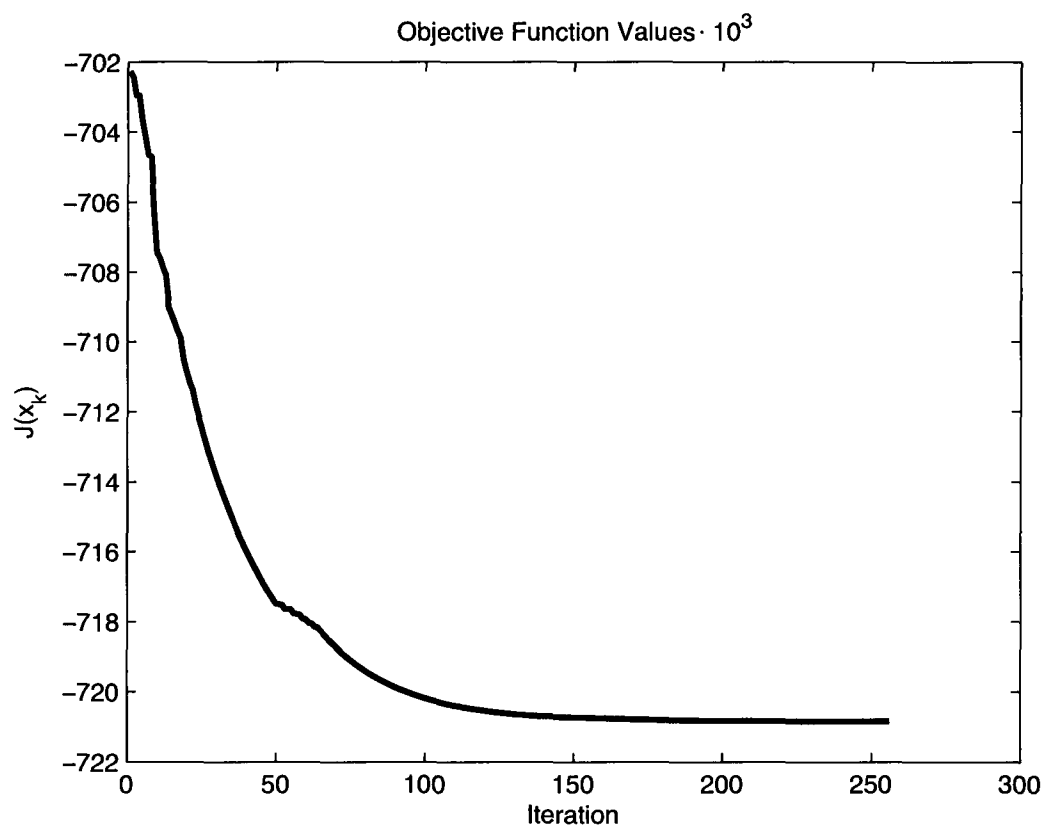


Figure 5.10: Objective Function Values - Experiment 2

The next figure plots the computed optimal well rates. As expected, injector 1 is the most active, followed by injector 3 and injector 2. Similar to the first experiment, the optimizer opened the injector, which is most likely to cause a water breakthrough, towards the end of the simulation to sweep the oil column towards the producer.

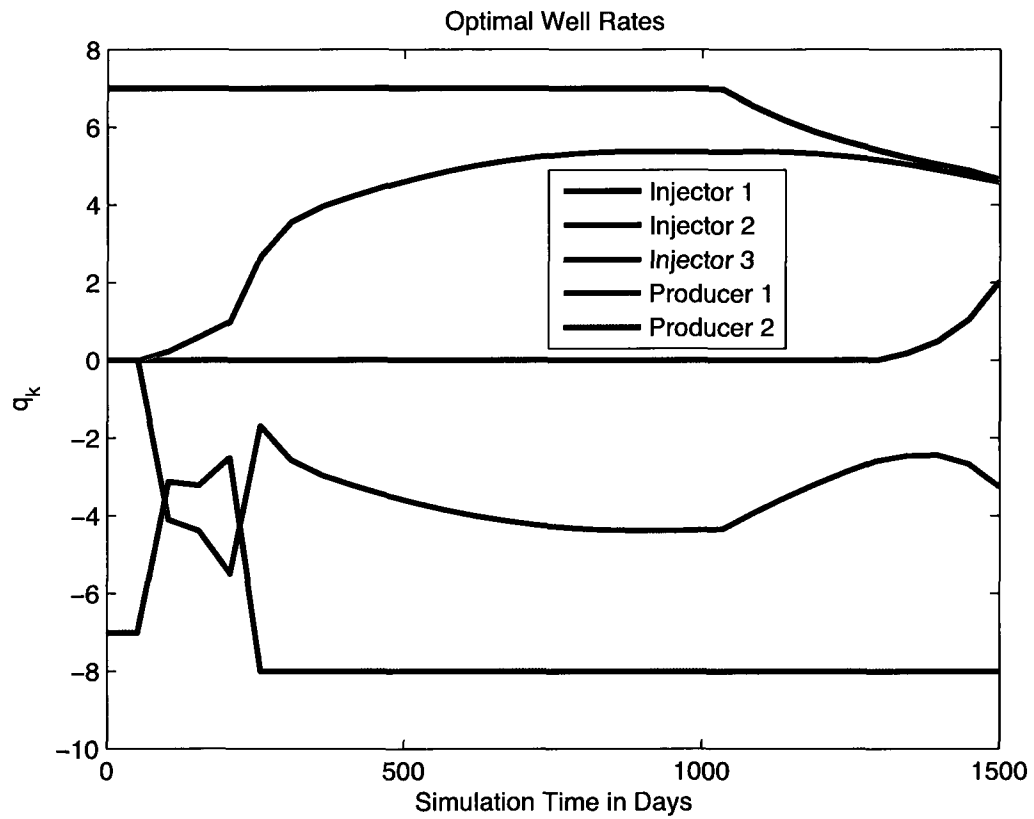


Figure 5.11: Computed Optimal Well Rates - Experiment 2

The last figure is a plot of the final aqueous saturation profile after 1,500 days of injection. The profile clearly depicts how the optimizer steered the fluids towards the two producers, avoiding water breakthrough.

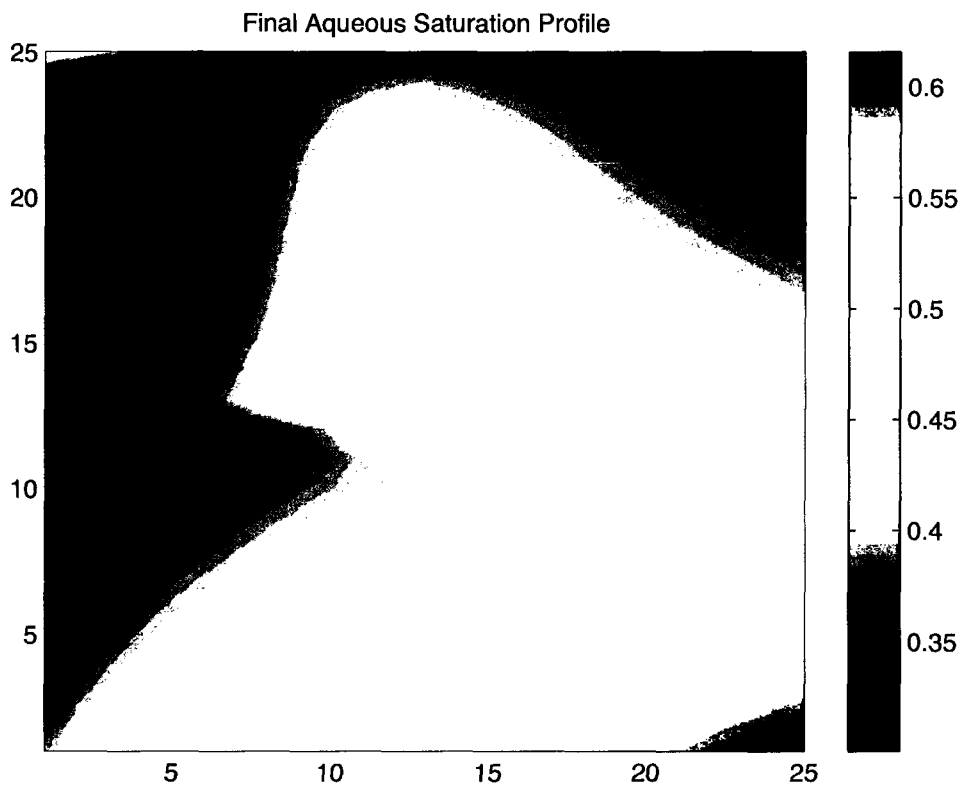


Figure 5.12: Final Aqueous Saturation Profile - Experiment 2

### 5.4.3 Experiment 3

The third experiment tests the optimization algorithm’s ability to handle large problems. The top layer of the SPE10 model consists of 13,200 simulation cells, arranged in a  $220 \times 60$  rectangular grid. I place a total of 4 injection and 4 production wells. Two of the injectors (I2, I3) are intentionally placed in close proximity to production wells (P1,P2). The expectation is that the optimizer computes an allocation pattern that avoids early water breakthrough by reducing these two injectors. The other two injectors (I1, I4) are placed as to sweep oil towards all 4 wells. Figure 5.13 depicts the well placement for the experiment, and the permeability distribution of the model.

The forward simulation time period for this experiment is 2,000 days with time steps of size  $\Delta t = 100$  days.

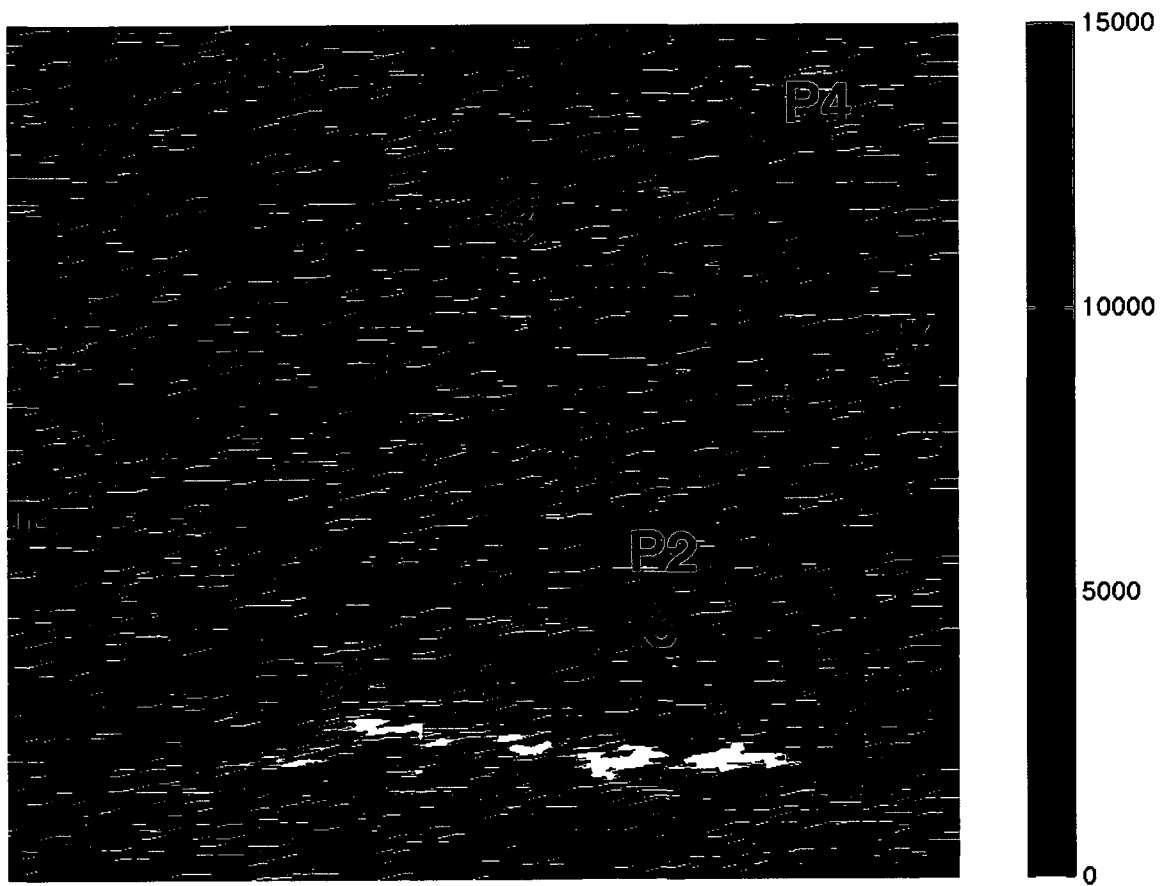


Figure 5.13: Optimization Experiment 3: Setup & Permeability Distribution

### Iteration Results

Following is the iteration information for both optimization algorithms. Due to the large number of iterations taken in the active set method, I show individual rows for the first 5 iterations, followed by increments of 10 iterations for the remainder of the optimization process.

Table 5.5: Iteration Results for Gradient Projection Algorithm - Experiment 3

Iteration	$f(x_k)$	$\ x_{k+1} - x_k\ _2$	$\ Z^T \nabla f(x_k)\ _2$	$ I(x_k) $
1	-1.1105e6	1.00e+0	1.76e2	0
2	-1.7766e6	1.86e-09	3.74e+1	38



Table 5.6: Iteration Results for Active Set Newton Algorithm - Experiment 3

Iteration	$f(x_k)$	$\ x_{k+1} - x_k\ _2$	$\ KKT\ _2$	$\max\{-\mu, 0\}$	$\max\{-\sigma, 0\}$	$ W_k $
1	-1.776e6	2.65e-2	1.95e+1	0	0	48
2	-1.776e6	4.24e+0	1.94e+1	3.40e+1	0	48
3	-1.792e6	2.38e-2	1.90e+1	0	0	48
4	-1.793e6	1.65e+0	1.89e+1	2.60e+1	0	48
5	-1.800e6	2.63e-2	2.26e+1	0	0	48
10	-1.822e6	1.67e-2	1.39e+1	0	0	49
20	-1.835e6	1.06e+0	1.34e+1	4.00e+0	0	46
30	-1.836e6	1.65e-2	1.34e+1	0	0	43
40	-1.837e6	1.61e-2	1.29e+1	0	0	43
50	-1.837e6	1.56e-2	1.23e+1	0	0	43
60	-1.838e6	1.53e-2	1.18e+1	0	0	43
70	-1.838e6	1.50e-2	1.13e+1	0	0	43
80	-1.838e6	1.45e-2	1.09e+1	0	0	43
85	-1.839e6	1.44e-2	1.07e+1	0	0	43

Following are figures that graph selected iteration information for the active set Newton algorithm.

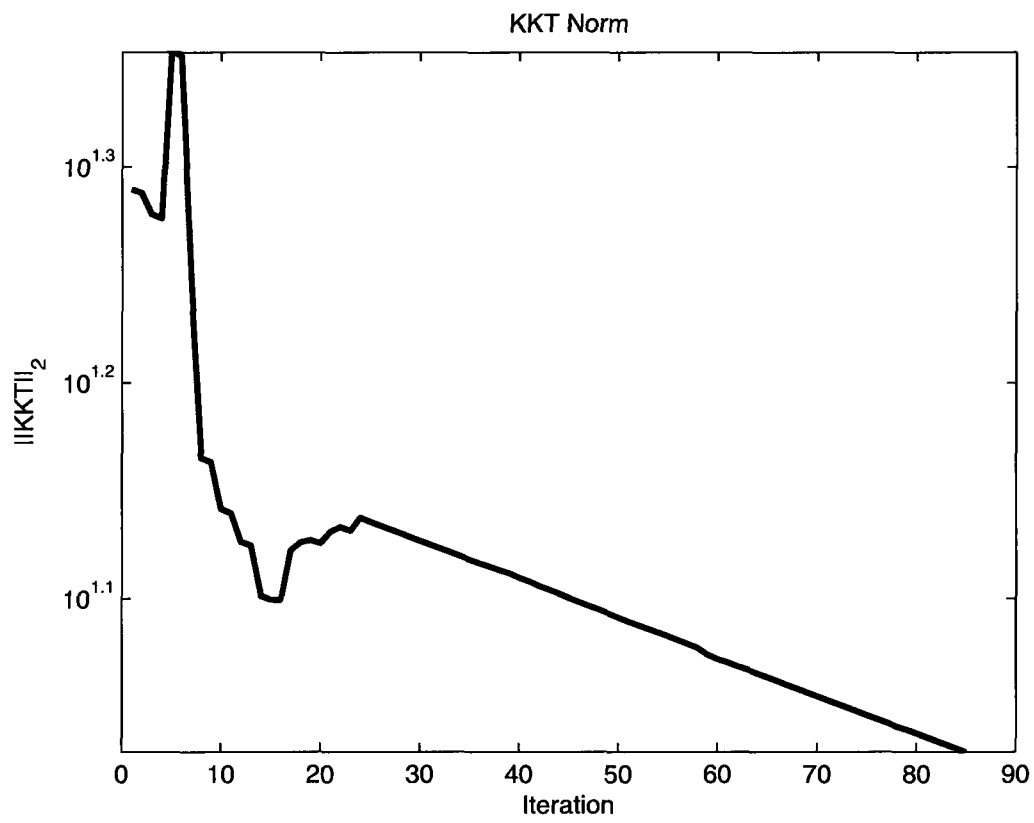


Figure 5.14: KKT Norm/Iteration - Experiment 3

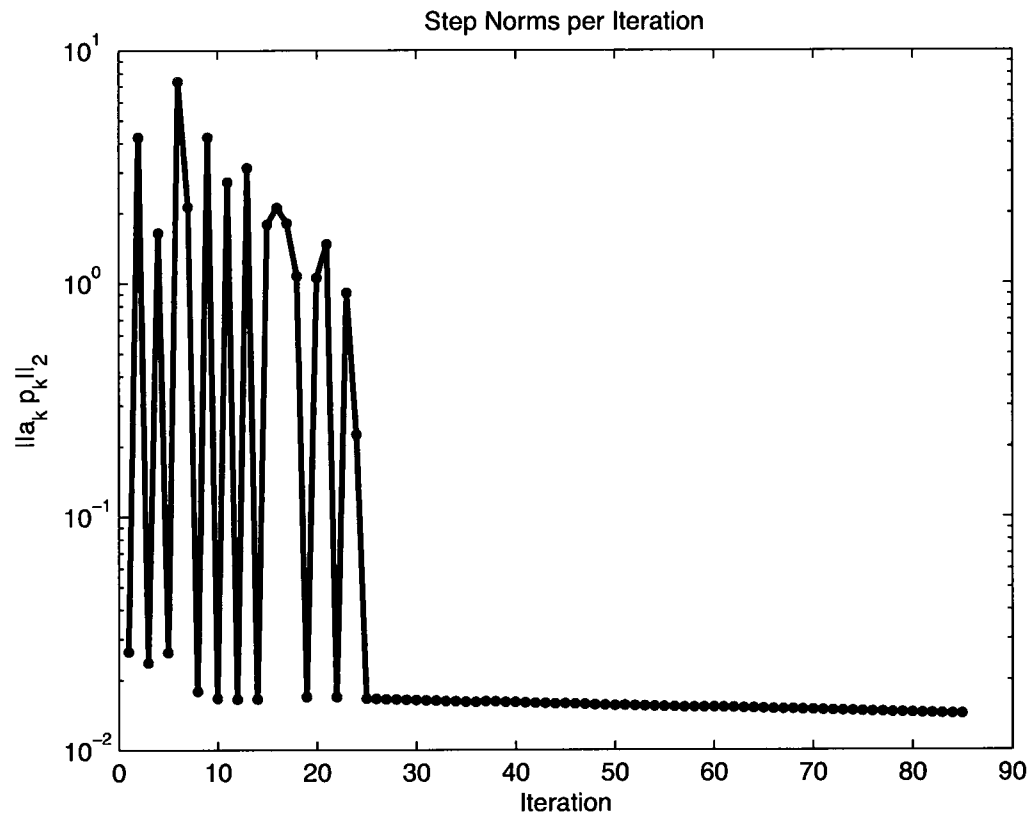


Figure 5.15: Step Norm/Iteration - Experiment 3

The following figure depicts the decrease of the objective function value during the iterations taken in the active set Newton method.

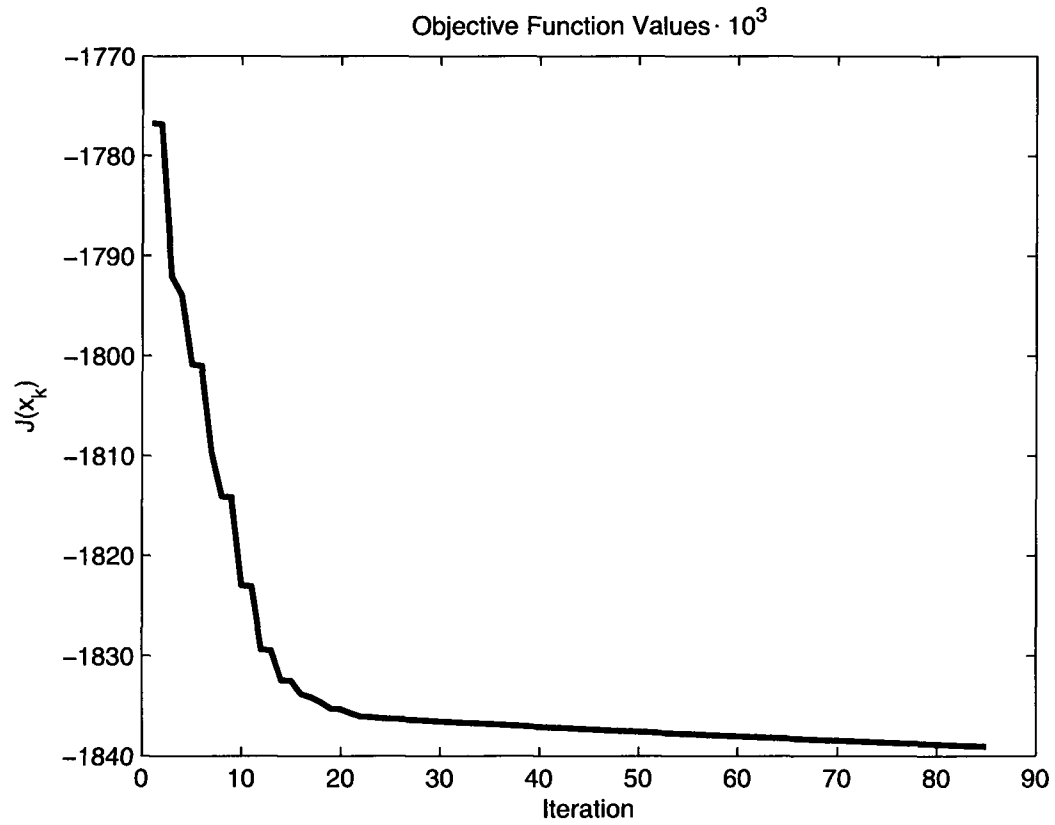


Figure 5.16: Objective Function Values - Experiment 3

The next figure plots the computed optimal well rates.

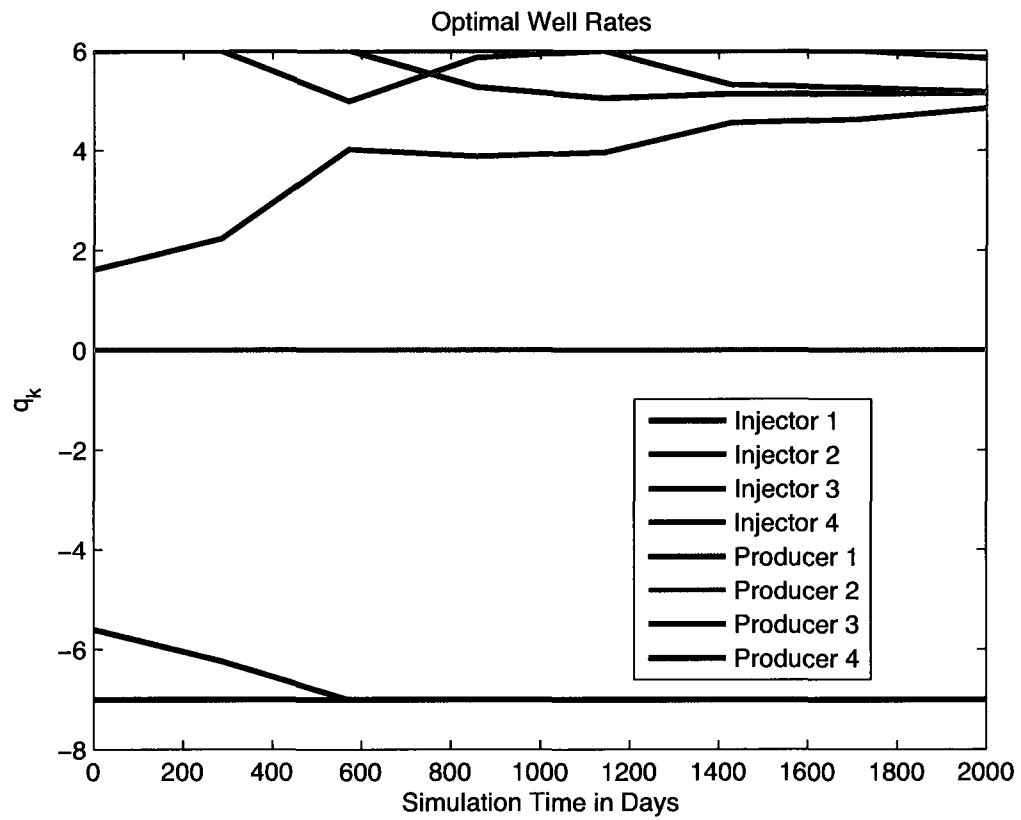


Figure 5.17: Computed Optimal Well Rates - Experiment 3

The last figure is a plot of the final aqueous saturation profile after 2000 days of injection.

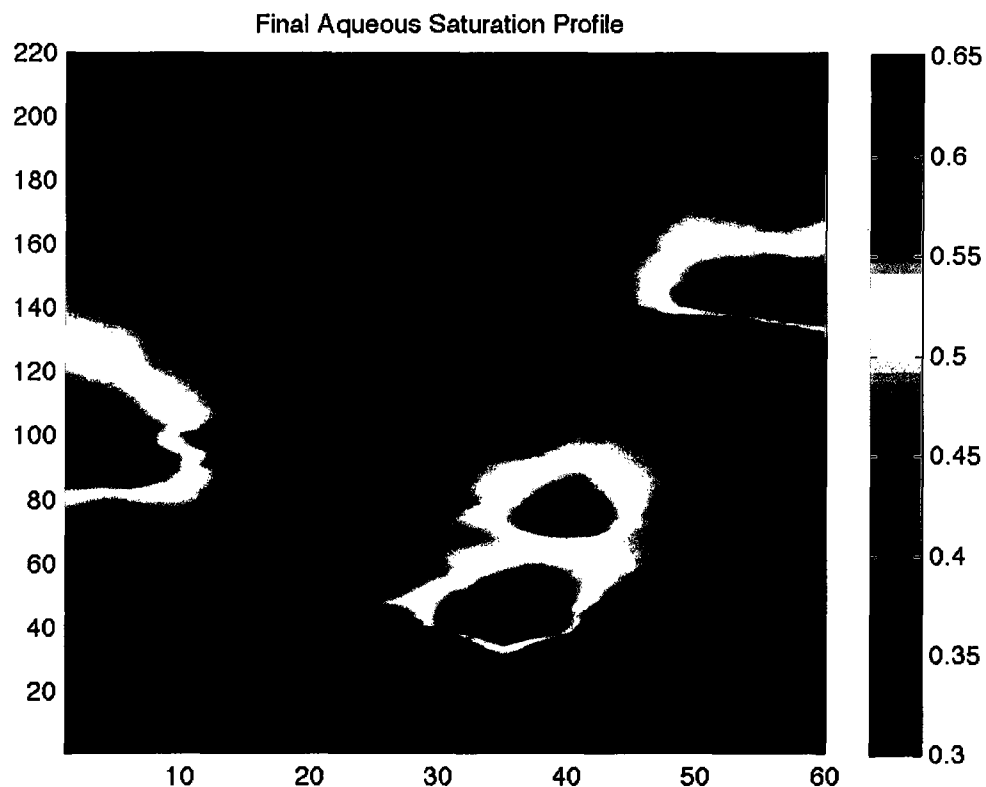


Figure 5.18: Final Aqueous Saturation Profile - Experiment 3

## 5.5 Discussion of Optimization Results

The optimization results show that the active set Newton method, baed on the approaches from Forsgren and Murray, is a robust and stable algorithm. With its ability to handle directions of negative curvature, and the approach to sweep off unwanted

active constraints, one can expect the algorithm to solve problems that are considered 'hard' for other nonlinear optimization codes. However, the fact that the algorithm adds new constraints at a very low rate (due to the linesearch strategy) is a potential drawback in situations with a large number of active constraints at the solution. For this reason, I combined the method with the gradient projection algorithm, to compute an initial guess with as many active constraints as possible. In case, that no degenerate constraints exist for the optimization problem, the algorithm is able to sweep of a multiple of unwanted active constraints at the same time. Hence, when coupled with the projected gradient method, both the problem of adding multiple constraints and deleting multiple constraints are addressed.

## Chapter 6

### Conclusions and Further Work

An adjoint based method for numerical reservoir optimization has been developed and successfully tested using three case studies. The components of the method (numerical reservoir simulator, adjoint computations, gradient computation, and Hessian-times-vector products) have been validated, and adjoints and gradients have been tested and compared against each other for two different time integration schemes, using varying time step sizes. The method is applicable to reservoir optimization problems with large state and control spaces.

In my future research, I need to address the extension of the method to numerical reservoir models with more complex physics, e.g. compressible fluid and rock models and mass transfer between miscible phases. An extension of the numerical simulator to support the compositional volume balance formulation needs to be considered. The known drawback of the second order active set Newton method is the slow



rate of adding new constraints to the working set. The implementation and test of alternative second order methods need to be considered. Furthermore, the use of piecewise constant well rates should be reconsidered and replaced with piecewise linear or piecewise quadratic controls. Last but not least, the handling of nonlinear constraints for both state and control variables needs to be researched in the context of more complex physics models for the numerical reservoir simulator.

# Bibliography

- [1] J. E. Aarnes, T. Gimse, and K.-A. Lie. An introduction to the numerics of flow in porous media using Matlab. In G. Hasle, K.-A. Lie, and E. Quak, editors, *Geometrical Modeling, Numerical Simulation and Optimisation: Industrial Mathematics at SINTEF*, pages 261–302, Heidelberg, 2007. Springer-Verlag.
- [2] I. Aitokhuehi and L. J. Durlofsky. Optimizing the performance of smart wells in complex reservoirs using continuously updated geological models. *Journal of Petroleum Science and Engineering*, 48:254–264, 2005.
- [3] A. H. Althuthali, D. Oyerinde, and A. Datta-Gupta. Optimal waterflood management using rate control. SPE 102478, Conference Paper, SPE Annual Technical Conference, San Antonio, Texas, U.S.A., September 2006.
- [4] W. Bangerth, H. Klie, M. F. Wheeler, P. L. Stoffa, and M. Sen. On optimization algorithms for the reservoir oil well placement problem. *Comp. Geosciences*, 10:303–319, 2006.

- [5] D. B. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Trans. Autom. Control*, (2):174–184, 1976.
- [6] A. Forsgren and W. Murray. Newton methods for large-scale linear equality-constrained minimization. *SIAM J. on Matrix Analysis and Applications*, 14:560–587, 1993.
- [7] A. Forsgren and W. Murray. Newton methods for large-scale linear inequality-constrained minimization. *SIAM J. Optim.*, 7(1):162–176, 1997.
- [8] M. Heinkenschloss. Numerical solution of implicitly constrained optimization problems. Technical Report TR08–05, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005–1892, 2008.
- [9] M.G. Ierapetritou, C.A. Floudas, S. Vasantharajan, and A.S. Cullick. Optimal location of vertical wells: Decomposition approach. Technical report, Dept. of Chemical Engineering, Princeton University and Strategic Research Center, Mobil Technology Co., Princeton, NJ 08544 and Dallas, TX 75244, 1997.
- [10] R.J. Lorentzen, A.M. Berg, G. Naevdal, and E.H. Vefring. A new approach for dynamic optimization of waterflooding problems. Conference Paper, SPE Intelligent Energy Conference, Amsterdam, The Netherlands, April 2006.
- [11] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Math. Programming*, 16(1):1–20, 1979.

- [12] K. W. Morton and T. Sonar. Finite volume methods for hyperbolic conservation laws. *Acta Numer.*, 16:155–238, 2007.
- [13] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Verlag, Berlin, Heidelberg, New York, 1999.
- [14] D. W. Peaceman. *Fundamentals of Numerical Reservoir Simulation*. Elsevier Science Inc., New York, NY, USA, 1977.
- [15] D. W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation. *Society of Pet. Eng. Journal*, 18(3):183–194, 1978.
- [16] D. W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation with nonsquare gridblocks and anisotropic permeability. *Society of Pet. Eng. Journal*, 23(3):531–543, 1983.
- [17] W. F. Ramirez. *Application of optimal control theory to enhanced oil recovery*. Elsevier Science Inc., New York, NY, USA, 1987.
- [18] G. W. Rosenwald and D. W. Green. A method for determining the optimum location of wells in a reservoir using mixed-integer programming. SPE 3981, Conference Paper, SPE-AIME 47th Annual Fall Meeting, San Antonio, Texas, U.S.A., October 1972.

- [19] R. Saito, G.N. de Castro, C. Mezzomo, and D.J. Schiozer. Value assessment for reservoir recovery optimization. *Journal of Petroleum Science and Engineering*, 32:151–158, 2001.
- [20] P. Sarma, L.J. Durlofsky, and K. Aziz. Efficient closed-loop production optimization under uncertainty. SPE 94241, Conference Paper, SPE Europec/EAGE Annual Conference, Madrid, Spain, June 2005.
- [21] D. Seifert, J. J. M. Lewis, C. Y. Hern, and N. C. T. Steel. Well placement optimization and risking using 3-d stochastic reservoir modeling techniques. Conference Paper, Congr Advanced wells and facilities - new and mature field applications, Stavanger, Norway, April 1996.
- [22] G.M. van Essen, M.J. Zandvliet, P.M.J. van den Hof, O.H. Bosgra, and J.D. Jansen. Robust optimization of oil reservoir flooding. Technical report, Delft Center for Systems and Control, Department of Geotechnology, Delft University of Technology, Mekelweg 2, 2628 CD Delft and Shell International Exploration and Production, Mijnbouwstraat 120, 2628 RX Delft, 1997.
- [23] P. Wang, M. Litvak, and K. Aziz. Optimization of production operations in petroleum fields. Conference Paper, SPE Annual Technical Conference, San Antonio, Texas, U.S.A, September, October 2002.
- [24] J. W. Watts. A compositional formulation of the pressure and saturation equations. *SPE Reservoir Engineering*, pages 243–252, 1986.

- [25] M.J. Zandvliet, O.H. Bosgra, van den Hof, J.D. Jansen, and J.F.B.M. Kraaijevanger. Bang-bang control in reservoir flooding. ECMOR A040, Conference Paper, 10th European Conference on the Mathematics of Oil Recovery, Amsterdam, The Netherlands, September 2006.